(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 October 2002 (03.10.2002)

PCT

(10) International Publication Number
WO 02/077759 A2

(51) International Patent Classification⁷: **G06F**

(21) International Application Number: PCT/US02/08293

(22) International Filing Date: 20 March 2002 (20.03.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/276,952	20 March 2001 (20.03.2001)	US
60/279,422	29 March 2001 (29.03.2001)	US
60/287,004	30 April 2001 (30.04.2001)	US
60/305,073	16 July 2001 (16.07.2001)	US
60/327,291	9 October 2001 (09.10.2001)	US

(71) Applicant *(for all designated States except US)*: **DEALIGENCE INC.** [US/US]; 30 Old Rudnick Lane, Dover, DE 19901 (US).

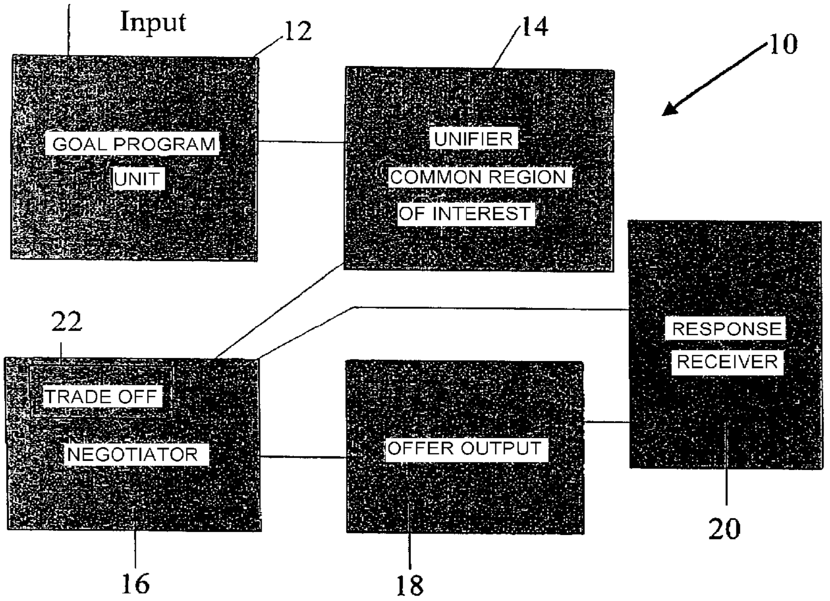
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **SHMUELI, Oded** [IL/IL]; 178 Hapisga Street, 36 001 Nofit (IL). **GOLANY, Boaz** [IL/IL]; 38 Harofe Street, 34 367 Haifa (IL). **SAYEGH, Robert** [IL/IL]; 63 Abas Street, 35 378 Haifa (IL). **SHACHNAI, Hadas** [IL/IL]; 12A Ehud Street, 34 551 Haifa (IL). **PERRY, Mordechal** [IL/IL]; 7/1 Snonit Street, P.O. Box 1804, 90 805 Mevasseret (IL). **GRADOVITCH, Noah** [IL/IL]; 10 Raul Wallenberg Street, 34 990 Haifa (IL). **YEHEZKEL, Benny** [IL/IL]; 74 Bialik Street, 52 441 Ramat Gan (IL).

(74) Agent: SHEINBEIN, Sol; G.E. Ehrlich (1995) Ltd., c/o Anthony Castorina, 2001 Jefferson Davis Highway, Suite 207, Arlington, VA 22202 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW,

[Continued on next page]

(54) Title: NEGOTIATING PLATFORM



(57) Abstract: A platform for supporting negotiation between parties to achieve an outcome, the platform comprising: a party goal program unit for: defining respective party's goal programs in respect of said outcome, said goal program comprising a plurality of objective functions and constraints associated with respective objective functions, for associating each of said objective functions with one of a plurality of levels of importance, and for assigning to objective functions within each level a respective importance weighting, said party goal program unit comprising a party input unit for allowing a party to provide data for a respective goal program, a goal program unifier, associated with said party goal program unit for receiving goal programs of respective parties, and carrying out unification of said goal programs by considering said objective functions objectivewise and levelwise with associated constraints in the respective goal programs to determine whether two goal programs have a common field of interest form which a mutually compatible outcome is derivable, a negotiator associated with said goal program unifier for receiving goal programs of respective parties, and carrying out negotiations using said goal programs by considering said objective functions objectivewise and levelwise with associated constraints in the respective goal programs to arrive at said mutually compatible outcome by carrying out minimization firstly objectivewise and then levelwise, therewith to form an offer, an output unit for offering said unified goal program to aid respective parties, and a response receiver for receiving from respective parties either counter offers or acceptance, said response receiver being operable to provide counter offers a new goal programs to said goal program negotiator for further unification.

WO 02/077759 A2

MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

without international search report and to be republished upon receipt of that report

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Negotiating Platform

Field of the Invention

The present invention relates to an apparatus, system or method for providing users with negotiation facilities, and more particularly but not exclusively to a general purpose electronic negotiation platform. The platform includes item choosing, ranking facilities, and deal splitting functionalities.

Background of the Invention

Numerous methods of carrying out negotiations are known, and many negotiation methods have been transferred to the electronic realm. The techniques provide various types of electronic deal making, including one-to-one negotiations, auctions, etc. The available art is generally dedicated to particular situations and is not general purpose. In particular the available art is dedicated to situations in which the principle or only negotiable variable is price, thus excluding whole realms of real negotiating life. For example, if a farmer wishes to negotiate with neighboring residents regarding land use issues pertaining to one of his fields, or if two sons are unable to agree a division of an inherited estate, then the current art is of only marginal help. Even if price is a factor, but only one of several factors of equivalent importance, then the present art is of only marginal help. Thus if a supermarket wishes to negotiate a contract for vegetables, then important factors are price, regularity and reliability of delivery, freshness and appearance. For a supplier, important factors are the quantity ordered and the price. The remaining factors are seen by him as limitations. None of the current art is able to provide a platform for a meaningful negotiation based on such a range of factors.

An example of a patent that allows variables other than price to be taken into account is US Patent 6,338,050, which discloses a multivariate negotiations engine for international transaction processing which: enables a sponsor to create and administer a community between participants such as buyers and sellers having similar interests; allows a buyer/participant to search and evaluate seller information, propose and negotiate orders and counteroffers that include all desired terms, request sample quantities, and track activity; allows a seller/participant to use remote authoring templates to create a complete Website for immediate integration and activation in the community, to evaluate proposed buyer orders and counteroffers, and to negotiate multiple variables such as prices, terms, conditions etc., iteratively with a buyer. The system provides secure databases, search engines, and other tools for use by the sponsor, which enable the sponsor to define the terms of community participation, establish standards, help promote the visibility of participating companies, monitor activity, collect fees, and promote successes. All this is done through a multivariate negotiations engine system operated at the system provider's Internet site, thus requiring no additional software at the sponsors', or participant sellers', or buyer's sites. This also allows buyers and sellers to use and negotiate payment options and methods that are accepted internationally. The system maintains internal databases that contain the history of all transactions in each community, so that sponsors, buyers and sellers may retrieve appropriate records to document each stage of interaction and negotiation. Documents are created by the system during the negotiation process.

It is noted that the above-described patent however, takes price as a principle feature and allows only subsequent iterative treatment for other features. Furthermore, the disclosure is specific to the one-on-one buyer seller model.

Summary of the Invention

According to a first aspect of the present invention there is thus provided

Claims

According to a first aspect of the present invention there is provided a platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit for:

defining respective party's goal programs in respect of said outcome, said goal program comprising a plurality of objective functions and constraints associated with respective objective functions,

for associating each of said objective functions with one of a plurality of levels of importance, and

for assigning to objective functions within each level a respective importance weighting, said party goal program unit comprising a party input unit for allowing a party to provide data for a respective goal program,

a goal program unifier, associated with said party goal program unit for receiving goal programs of respective parties, and carrying out unification of said goal programs by considering said objective functions objectivewise and levelwise with associated constraints in the respective goal programs to determine whether two goal programs have a common field of interest from which a mutually compatible outcome is derivable,

a negotiator associated with said goal program unifier for receiving goal programs of respective parties, and carrying out negotiations using said goal programs by considering said objective functions objectivewise and levelwise with associated constraints in the respective goal programs to arrive at said mutually compatible outcome by carrying out minimization firstly objectivewise and then levelwise, therewith to form an offer,

an output unit for offering said unified goal program to said respective parties,

a response receiver for receiving from respective parties either counter offers or acceptances, said response receiver being operable to provide counter offers as new goal programs to said goal program negotiator for further unification.

Preferably, said negotiator comprises a trade-off unit for identifying excludable objectives levelwise in a first party's goal program for conditional weakening from said outcome in a trade-off involving strengthening of other objectives within the same level of said first party.

Preferably, said negotiator comprises a trade-off unit for identifying excludable objectives levelwise in a first party's goal program for conditional weakening from said outcome in a trade-off involving weakening of other objectives within the same level of another party.

Preferably, said party goal program unit is operable to express said objective functions in a weighted hierarchy according to the respective associated level of importance, and to express each constraint in terms of a constraint variable, thereby to form an expression for minimization at said negotiator.

Preferably, said party goal program unit is operable to use data from said party data input unit to apply coefficients to said constraint variables.

Preferably, said party goal program unit is operable to apply user input to provide different values of coefficients to said constraint variables for deviations of a corresponding objective in respective positive and negative directions.

Preferably, said objective program unit is operable to apply said user input to apply said coefficient values to define any one of a group comprising:

a strong one sided objective,

a weak one sided objective,

a complex single sided objective,

a simple two sided objective,

a complex two sided objective,

a simple first side-complex second side objective,

a simple two-sided objective with an indifferent range,

a complex two sided objective with an indifferent range, and

a simple first side-complex second side simple objective with an indifferent range.

Preferably, at least one objective comprises a series of discrete values.

Preferably, said party goal program unit is operable to apply user input to formulate weightings for respective ones of said discrete values, thereby to express a preference between said discrete values.

Preferably, at least one objective function comprises a continuous variable, and wherein said party goal program unit is operable to apply user input to determine whether said continuous variable is to be maximized or to be minimized.

Preferably, said negotiator is operable to arrange said levels in a hierarchy and to carry out minimization by summing objective functions associated with constraint variables levelwise in said hierarchy.

Preferably, said negotiator is operable to carry out minimization by carrying out minimization per constraint variable and setting a maximum bound per deviation.

Preferably, said negotiator is operable to carry out minimization for an expression comprising first ones of said constraint variables and then to add further constraint variables to said expression for a further minimization stage.

Preferably, said party input unit is configured to receive data from a user interface.

Preferably, said party input unit is configured to receive data from a software agent.

Preferably, said party input unit is operable to identify parameter data missing from an input and further comprises a default value generator for generating said missing parameter.

Preferably, said party input unit is operable to identify parameter data missing from an input and further comprises a default register of values for expected parameters.

Preferably, said party input unit is operable to obtain lower and upper bounds for at least some of said objective functions.

Preferably, goal program unit is operable to use said upper and lower bounds to express deviations of a respective objective function from a target value relatively, thereby to render said deviations subject to comparison by said unifier or by said negotiator.

Preferably, said party input unit is operable to obtain a objective function interval, and a value for a penalty for deviating from said interval, and wherein said unifier is operable to define a working interval between two objective functions as an intersection between respective intervals.

Preferably, said unifier is operable to determine that a target value of one of said objective functions is outside said working interval, and to modify said target value to approach a closest boundary of said working interval.

Preferably, said unifier is operable to apportion said penalty in accordance with said target value modification.

Preferably, said intersection is a point.

Preferably, said party goal program unit comprises operability for determining that an intersection is small to the satisfaction of respective parties and, when said intersection is recognized as small, said negotiator is operable to select a point within said intersection being a midpoint between respective target values.

Preferably, said negotiator is operable to measure deviations within said interval as a fraction of a total size of said interval.

Preferably, said party goal program unit is operable to obtain importance values for deviations from said target and wherein said negotiator is operable to use said importance value as a multiplier to measure said deviation.

Preferably, said negotiator is operable to identify intersections that are small and distant from a target value compared to one of said objective functions and large and inclusive of a target value compared to another of said objective functions, to set an effective target at the closest intersection boundary and to set a transformed deviation as giving the arithmetic deviation when multiplied by the effective target and then added to the difference between the old target and the effective target, to produce a result which is divided by the old target.

Preferably, said party input unit is operable to permit a party to define at least one single dimension interval objective in respect of said outcome, and to associate said objective with a range of indifference having an upper bound and a lower bound, a first weighting value for deviations below said lower bound, a second weighting value for deviations above said upper bound and a relative importance for said objective,

said unifier being operable to use said range of indifference, said weightings and said relative importance to unify said at least one objective with at least one other objectives to determine said compatibility.

Preferably, said at least one other objective is a corresponding objective in a goal program of an opponent.

Preferably, said party input unit further comprises a prioritizer for allowing said respective party to define said association of a respective objective function levelwise with other objectives, said party input unit further being operable to allow said party to establish a priority with objectives within a same level.

Preferably, said party input unit is operable to permit a party to define a two dimensional trade-off objective by entering two two-dimensional points, said party goal program unit being operable to define a trade-off line between said two points.

Preferably, said party input unit is operable to permit a party to define weights for deviation from said trade-off line.

Preferably, said party input unit is operable to permit a party to define a relative importance for said two dimensional trade-off objective.

Preferably, said party input unit is operable to permit a party to associate said two dimensional trade-off objective levelwise with other objectives.

Preferably, party enterable weightings are usable by said unifier to establish a priority between objectives in a same level.

Preferably, said party input unit is operable to allow a party to define at least one single dimension two-point objective in respect of said outcome, and to associate said objective with an upper point of preference, and a lower point of preference, a first weighting value for deviations below said lower point of preference, and a second weighting value for deviations above said upper point of preference, said goal program unit being operable to provide weightings to a region included between said points of preference by assigning said first weighting value below said upper point of preference and said second weighting value above said lower point of preference and defining an overall weighting within said region as a minimum of said weighting values,

and wherein said unifier is operable to use said range of indifference, said weightings, and said minimum to unify said at least one objective with other objectives to determine said compatibility.

Preferably, said party input unit is operable to permit a party to define a relative importance for said single dimensional two point objective.

Preferably, said party input unit is operable to permit a party to associate said single dimensional two point objective levelwise with other objectives.

Preferably, said relative importance is usable by said negotiator to establish a priority between objectives in a same level.

Preferably, said party input unit is operable to permit a user to define a piecewise linear two-dimensional goal by entering at least three two-dimensional points, said party goal program unit being operable to define a trade-off line between said three points, said negotiator being operable to apply penalty values to points on said trade-off line in accordance with their distances from said points.

Preferably, said party input unit is operable to permit a user to define a first deviation weight for deviating in a first direction from said trade-off line and a second deviation weight for deviating in a second direction therefrom.

Preferably, said party input unit is operable to permit parties to define objectives comprising pairwise trade-offs having at least two points and a trade-off therebetween, and to associate each of said trade-offs with a slope.

Preferably, said party goal program unit is operable to prevent inconsistent trade-offs to be defined within the platform by preventing said party input unit from accepting more than one trade-off from referring to any given point.

Preferably, said party goal program unit is operable to warn users of inconsistent trade-offs by outputting a warning whenever a trade-off being entered has a point already included in a previously entered trade-off.

Preferably, said party input unit is further operable to allow a party to define disjunctive constraints in respect of objectives, said goal program unit comprising a disjunctive constraint processor for translating a disjunctive expression into at least one linear conjunctive expression, and wherein said unifier is operable to utilize said linear conjunctive expression to unify said at least one objective with other objectives to determine said compatibility.

Preferably, said disjunctive expression comprises a series of relationships including equality relationships.

Preferably, said disjunctive constraint processor is operable to carry out said translation by expressing at least one of said equality relationships as the union of two corresponding inequalities that meet at a point of equality of said equality relationship.

Preferably, said disjunctive constraint processor is operable to define binary variables for said relationships, for setting wherever said relationships are satisfied, and wherein said negotiator is operable to sum said variables to determine a satisfaction level for said objective.

Preferably, said party goal program unit is operable to set a requirement of a minimum number of satisfied relationships for use by said negotiator.

Preferably, said party input unit is further operable to permit each party to define weighting values for a discrete variable predefined per outcome, for use in said goal program definition, and

said negotiator being operable to carry out negotiation of said goal programs by considering said weighting values to arrive at an outcome comprising an offered one of said values.

Preferably, said party goal program unit is operable to use said weightings for respective values of said discrete variable to arrange said variable as a weighted hierarchy, said hierarchy being usable by said negotiator to arrive at said offered one of said discrete values.

Preferably, said party goal program unit is operable to use said values and respective weightings to build summation functions, therefrom to express said variable in quantitative manner.

Preferably, said negotiator is operable to attempt offer formation by setting one of said discrete values to one and the remainder to zero, and then to calculate said summation functions to contribute to a fulfillment level of each goal.

Preferably, said negotiator is operable to reattempt offer formation by setting different ones of said discrete values to one, thereby to find a value which maximizes said fulfillment level.

Preferably, said party goal program unit is operable to represent date information as accumulated minutes from a threshold starting date, and further to modify said dates relative to upper and lower bounds entered via said party input unit.

Preferably, said party input unit is further operable to allow input of variables in association with said objective functions and a linkage between a first and a second of said variables, said linkage defining a trade-off path of deviations with respect to said target values, said negotiator being operable to use said series of variables including said trade-off path to negotiate an outcome in respect of said at least one objective with other objectives, thereby to arrive at formation of an offer.

Preferably, said party goal program unit is operable to express said second variable as a function of said first variable, thereby to represent said linkage, and wherein said negotiator is operable to represent deviations from respective target values as deviations from the target value of said first variable.

Preferably, said party goal program unit is operable to express said trade-off as separate deviation variables in respect of said first variable and in respect of said second variable wherein said separate deviation variables are orthogonal.

Preferably, said party input unit is operable to permit an association of a relative importance level to said trade-off.

Preferably, said party goal program unit is operable to calculate a relative importance level for said trade-off as an average of respective relative importance levels of said first and second variables.

Preferably, said unifier comprises a goal program generalizer to form a generalization of received goal programs for use in said unification.

Preferably, said party goal program unit is operable to translate said input received from said party input unit into said objective functions and constraints on said objective functions within said goal program, said negotiator comprising an optimizer to find best values for said objective functions and constraints, therewith to obtain a best solution for the goal program for output as a first offer, and then iteratively to produce further solutions until an offer is accepted, thereby to achieve said outcome.

Preferably, said negotiator comprises a percentage reducer for taking ones of said objective functions or levels in turn, worsening them by a predetermined percentage, thereby to produce said further solutions.

Preferably, the percentage reducer is settable to take each of said objective functions in turn beginning with a most important objective within each level, until a solution is accepted.

Preferably, said objective functions are arranged in levels and said percentage reducer is settable to take objective functions of a first of said levels only.

Preferably, said negotiator comprises a worst case calculator for determining a worst case level for ones of said objective functions and constraints, thereby to obtain a worst acceptable offer.

Preferably, said goal program is arranged as pairwise bounded variables, and said negotiator comprises an arbitrary case calculator for taking one of each pair of variables, setting it to an arbitrary value within its respective bounds, taking the other of said pair of variables and setting it to zero, therefrom to calculate ones of said iterative solutions.

Preferably, said negotiator further comprises an average case calculator, associated with said optimizer and with said worst case calculator, for

taking said best solution and said worst solution, each solution having corresponding values,

associating ones of said corresponding values, and

constraining variables of said goal program towards an average of each of said corresponding values, therewith to provide an average solution.

Preferably, said goal program objective functions are arranged levelwise and said average case calculator is operable to carry out said associating and said constraining successively levelwise, thereby to produce said series of iterative solutions.

Preferably, said negotiator comprises a solution sorter for comparing goal program solutions by solving said goal program for each one of a series of solutions and ranking the solutions, said negotiator being operable to use said ranking to apply preference to different solutions.

Preferably, wherein said negotiator further comprises a thresholder associated with said solution sorter for applying a threshold to said evaluations to exclude ones of said series of solutions.

Preferably, said solution sorter further comprises a solution completer for applying best values to incompleted variables in incomplete ones of said solutions, thereby to allow said goal program to be evaluated for said incomplete solutions.

Preferably, said solution sorter is settable to find the best solution from said series of solutions by identifying the highest ranked solution and discarding the remaining solutions.

Preferably, said solution sorter comprises a memory, set to hold a predetermined number of solutions, and a comparator to compare a new solution with each solution in said memory, and further comprising a control unit for adding said new solution to said memory if its evaluation is larger than any solution in said memory, and for discarding a lowest ranked solution in said memory.

Preferably, said goal program unit comprises a data input unit for receiving user defined output values, and wherein said goal program unit is operable to set said output values as single value constraints and to flag said constraints as unchangeable for said unifier.

Preferably, said data input unit is operable to output an error indicator if said single value constraints render said goal program insoluble.

Preferably, the unifier further comprises a goal program input unit for receiving a local party's goal program and an opponent's goal program to be unified therewith, said goal programs comprising objective functions associated with constraints and being arranged in levels, and the negotiator further comprises:

an optimizer for finding best solutions to goal programs, connected to find best values for said objective functions and constraints of said local party's goal program levelwise, and

a worst case calculator for finding worst solutions for goal programs, connected to find worst values for said objective functions and constraints of said opponent's goal program levelwise, said negotiator being operable to:

use said optimizer and said worst case calculator in succession level by level to produce successive value sets for evaluation therefrom to form level by level unification offers, and

advance from one level to another level only following acceptance by said parties of a unification offer regarding a previous level.

Preferably, said negotiator comprises a constraint updater for updating constraints upon advance from one level to another level in accordance with said respective acceptance.

Preferably, said negotiator further comprises an offer improver operable to provide an improved offer by making a change in a selected one of said variables to bring about an improvement in the evaluation of the opponent's goal program.

Preferably, said change in said variable is calculated such that said improvement is a predetermined proportion of a difference between a previous offer made and a best possible evaluation made for the opponent.

Preferably, said offer improver is operable to use a value of said selected variable in a last opponent's offer to moderate said change.

Preferably, said offer improver is operable to calculate a protection value, and to use said protection value to limit a reduction in the evaluation of the local party's goal program as a consequence of said improvement to the opponent's goal program evaluation.

Preferably, said protection value is a proportion of the difference between a worst case evaluation of the local party's goal program and an evaluation of last previous offer thereof.

The embodiment further comprises an offer improver for taking goal program values of a previous local party offer and one value in turn from a previous opponent offer, testing the opponent value against local constraints, and if it fits within the constraints then substituting it into the previous local party offer thereby to provide an improved offer.

Preferably, the negotiator further comprises a goal program input unit for receiving a local party's goal program, said goal programs comprising objective functions associated with constraints and being arranged in levels, and said negotiator further comprises:

an optimizer for finding best solutions to goal programs, connected to find best values for said objective functions and constraints of said local party's goal program levelwise, and

a stay close processor for determining variable improvement directions from monitoring of received offers from said opponent and carrying out value perturbations in said directions,

said negotiator being operable to:

use said optimizer to produce a first offer for a first level,

to advance from one level to another level only following acceptance by said parties of an offer regarding a previous level, and use said stay close processor to produce a first offer for each subsequent level, thereby to arrive at said outcome.

Preferably, said negotiator comprises a constraint updater for updating constraints upon advance from one level to another level in accordance with said respective acceptance.

Preferably, said negotiator further comprises

a gap value determiner, for determining a gap for use in offer improvement, and

a value improver, associated with said gap value determiner, for inserting a predetermined proportion of said gap as a constraint of said goal program, and wherein said stay close processor is associated with said value improver thereby to apply predetermined gap proportion in said direction to provide an improved offer.

Preferably, said stay close processor is operable to monitor two successive opponent offers for value changes therebetween, and to assign to each respective changing variable a weight for use in providing an improved offer, the magnitude of said weight being selected in accordance with a monitored relative size of a corresponding value change of said opponent.

Preferably, said gap is a constant.

Preferably, said constant is a difference between a best value and a worst value of a corresponding variable.

Preferably, said gap is a difference between a last local proposal and a last opponent proposal.

An embodiment further comprises a negotiation necessity tester, associated with said unifier, for joint solving of said local and said other goal program to form a joint goal program comprising optimal solutions for each of said local and said other goal program, said negotiation necessity tester being set to determine whether there lies a single solution that includes both optimal solutions within said common ground, and if so, to inhibit passing of said goal programs to said negotiator.

An embodiment further comprises a mediation unit callable by both parties levelwise during said negotiations, said mediation unit being operable to retain agreed objectives of previously solved levels and to apply a summation formula to solve a current level.

Preferably, said summation formula is:

$$\left\{\frac{\sum w_k^{+} \cdot \delta_k^{+} + \sum w_k^{-} \cdot \delta_k^{-}}{\sum w_k^{+} + \sum w_k^{-}}\right\} + \left\{\frac{\sum v_k^{+} \cdot \gamma_k^{+} + \sum v_k \cdot \gamma_k^{-}}{\sum v_k^{+} + \sum v_k^{-}}\right\}$$

The mediation unit is preferably callable by both parties during said negotiations, said mediation unit being operable to stop operation of said negotiator, apply a summation formula to provide a median solution between respective goal programs, and to provide said median solution as an offer to both parties.

Preferably, each goal program is expressed as a series of decision variables each having an upper bound, a lower bound, a target value and one or more constraints, the platform further comprising a form offer unit for providing a form offer to the parties, the unit being operable to assign to each of the goal programs a weighting such that the sum of the weightings is unity, for each variable and to calculate the offer by minimizing relative deviations for each variable over said goal programs weighted according to said assigned weightings.

A discrete variable form offer unit is preferably provided to transform values of said discrete variable into a continuous domain, to carry out minimization in light of goal program objectives of said two parties in said continuous domain, and to transform the minimization results back to discrete values, thereby to provide a form offer.

Preferably, said negotiator is operable to provide a value for at least one objective by comparing said at least one objective from a local goal program against the entirety of an opponent goal program.

The platform preferably comprises an item catalog for storing a plurality of items for offer in terms of values of said objectives, wherein said negotiator is operable to provide offers in terms of nearest items in said catalog.

Preferably, said negotiator comprises:

an item manager for recursively determining which items of said catalog are currently within the scope of negotiations,

a first round manager, associated with said item manager, for managing levelwise goal program negotiation to successively reduce the number of said items within said scope to a predetermined threshold number of items,

a second round manager, associated with said item manager, for managing levelwise program negotiation to produce successive offers, and

an item associator, connected to said second round manager and to said item manager, for expressing said successive offers in terms of items within said scope.

Preferably, said item manager is operable to measure distance from said scope in terms of an opponent goal program.

Preferably, said item manager is operable to measure distance from said scope in terms of a joint goal program.

Preferably, said item manager is operable to measure distance initially in terms of a local goal program, to order said items and iteratively to remove most distant items.

Preferably, one of said objectives is compatibility with a second item.

The platform further comprises an offer delay timer, associated with said negotiator, for introducing determinable delays between issuance of successive offers to an opponent.

Preferably, said offer delay timer is operable to set successively increasing delays.

Preferably, said offer delay timer is operable to set delays to change levelwise.

Preferably, a magnitude of said delay is based on a relative change between succeeding opponent offers.

Preferably, said offer delay timer is operable to set user determined delays.

Preferably, at least one of said objectives comprises a dynamic variable.

Preferably, at least some of said constraints are associated with dynamically changing variables.

According to a second aspect of the invention there is provided a platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit for allowing each party to define a plurality of goals in respect of said outcome, and to associate each of said goals with a respective level of importance, therefrom to form for each party a goal program,

said party input unit being operable to obtain a target value and upper and lower bounds for at least one of said goals, said party goal program unit being operable to use said upper and lower bounds to express deviations from said target values in relative terms, thereby to render deviations from different goals comparable.

According to a third aspect of the present invention there is provided a platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit for allowing a party to define a plurality of goals in respect of said outcome, and to associate at least one of said goals with a target value, an acceptable interval, and a penalty for deviation from said interval, and

a unifier, for determining common ground between said goal program and at least one other goal program,

and a negotiator, operable to form offers within said common ground by mutual quantifying of a objective function of said at least one goal program with objective function of said at least one other goal program having a target value and an interval, by determining an intersection between said intervals and if said target value is outside said intersection then moving said target value by a deviation amount to a closest boundary of said intersection, said negotiator further being operable to apportion said penalty for deviation amount in accordance with an extent of said deviation of said target value.

According to a third aspect of the present invention there is provided a platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit for allowing a party to define at least one goal program having a plurality of objectives in respect of said outcome, and to associate at least one of said objectives with a range of indifference having an upper bound and a lower bound, a first weighting value for deviations below said lower bound, a second weighting value for deviations above said upper bound and a relative importance for said objective,

and a negotiator, associated with said goal program unit, said negotiator being operable to use said range of indifference, said weightings and said relative importance to obtain an outcome for said at least one objective in view of other objectives, by producing successive offers.

Preferably, said party input unit further comprises a prioritizer for allowing said objective to be associated levelwise with other objectives.

According to a fourth aspect of the present invention there is provided a platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit operable to permit a party to define a two dimensional trade-off objective by entering two two-dimensional points, said party goal program unit being operable to define a trade-off line between said two points, and a negotiator, associated with said goal program unit, said negotiator being operable to use said trade-off line to unify said at least one objective with other objectives to arrive at said outcome via a series of successive offers.

147.According to a fifth aspect of the present invention there is provided a platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit for allowing a party to define at least one single dimension two-point objective in respect of said outcome, and to associate said objective with an upper point of preference, and a lower point of preference, a first weighting value for deviations below said lower point of preference, and a second weighting value for deviations above said upper point of preference, said goal program unit being operable to provide weightings to a region included between said points of preference by assigning said first weighting value below said upper point of preference and said second weighting value above said lower point of preference and defining an overall weighting within said region as a minimum of said weighting values,

and a negotiator, associated with said goal program unit, said negotiator being operable to use said range of indifference, said weightings, and said minimum to converge said at least one objective with other objectives to arrive at successive offers to achieve said outcome.

According to a sixth aspect of the present invention there is provided a platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit operable to permit parties to define objectives comprising pairwise trade-offs having at least two points and a trade-off therebetween, and to associate each of said trade-offs with an inclination value, wherein said party goal program unit is operable to prevent inconsistent inclination values to be defined within the platform by preventing said party input unit from accepting more than one trade-off that refers to a same point

and a negotiator for negotiating with other parties via goal programs to achieve an outcome consistent with said objectives.

According to a seventh aspect of the present invention there is provided a platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit operable to permit parties to define objectives comprising pairwise trade-offs having at least two points and a trade-off therebetween, and to associate each of said trade-offs with an inclination value, wherein said party goal program unit is operable to warn users of inconsistent inclination values by outputting a warning whenever a trade-off being entered has a point already included in a previously entered trade-off, and

a negotiator for negotiating with other goal programs to achieve an outcome consistent with said objectives.

According to an eighth aspect of the present invention there is provided a platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit for allowing a party to define at least one objective in respect of said outcome, and to associate said objective with a series of variables including disjunctive constraints, said goal program unit comprising a disjunctive constraint processor for translating a disjunctive expression into at least one linear conjunctive expression,

and a negotiator, associated with said goal program unit, said negotiator being operable to use said series of variables including said linear conjunctive expression to negotiate an outcome consistent with said goal program and with other goal programs.

According to a ninth aspect of the present invention there is provided a platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit for allowing each party to define a plurality of objectives in respect of said outcome, each objective comprising a plurality of variables, therefrom to form for each party a goal program, wherein a variable having discrete values is predefined, and wherein said party input unit is operable to receive discrete values of said variable for use in said objective definition,

a unifier, associated with said party goal program unit for receiving goal programs of respective parties, said objectives including discrete values of said variable, said unifier being operable to carry out unification of said goal programs by considering said discrete values to arrive at a common region of said discrete variables amongst said goal programs, and

a negotiator operable to utilize fulfillment levels associated with said discrete values to produce successive offers to converge on an outcome within said common region..

According to a tenth aspect of the present invention there is provided a platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit for allowing a party to define at least one objective in respect of said outcome, and to associate said objective with at least two variables each having a target value, said party input unit further allowing input of a linkage between a first and a second of said variables, said linkage defining a trade-off path of deviations with respect to said target values,

a unifier, associated with said goal program unit, said unifier being operable to use said series of variables to unify said at least one objective with other objectives to find a common area of interest, and

a negotiator, associated with said unifier, for using said trade-off path to find a mutually acceptable outcome within said common area.

According to an eleventh aspect of the present invention there is provided a platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit for defining goal programs in respect of an outcome, the goal program unit comprising a party input unit for allowing a party to input data relating to said goal program, said goal program unit being operable to translate said values into objective functions and constraints on said objective functions within said goal program,

and a negotiator, associated with said goal program unit, said negotiator comprising an optimizer to find best values for said objective functions and constraints, therewith to obtain a best solution for the goal program for output as a first offer, and then iteratively to produce further solutions until an offer is accepted, thereby to achieve said outcome.

According to a twelfth aspect of the present invention there is provided a platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit for defining goal programs in respect of an outcome, the goal program unit comprising a party input unit for allowing a party to input values, said goal program unit being operable to translate said values into objective functions and constraints on said objective functions within said goal program,

and a negotiator, comprising a solution sorter for comparing goal program solutions by evaluation of said goal program for each one of a series of solutions and ranking the solutions according to said evaluations, said negotiator being operable to use said ranking to apply preference to different solutions.

According to a thirteenth aspect of the present invention there is provided a platform for supporting negotiation between a local party and an opponent party to achieve an outcome, the platform comprising a unifier, the unifier comprising:

a goal program input unit for receiving a local party's goal program and an opponent's goal program to be unified, said goal programs comprising objective functions associated with constraints and being arranged in levels,

an optimizer for finding best solutions to goal programs, connected to find best values for said objective functions and constraints of said local party's goal program levelwise, and a worst case calculator for finding worst solutions for goal programs, connected to find worst values for said objective functions and constraints of said opponent's goal program levelwise,

said negotiator being operable to:

use said optimizer and said worst case calculator in succession level by level to produce successive best local and worst opponent value sets for evaluation therefrom to form level by level offers, and

to advance from one level to another level only following acceptance by said parties of a offer regarding a previous level.

According to a fourteenth aspect of the present invention there is provided a platform for supporting negotiation between a local party and an opponent party to achieve an outcome, the platform comprising a negotiator, the negotiator comprising:

a goal program input unit for receiving a local party's goal program, said goal programs comprising objective functions associated with constraints and being arranged in levels,

an optimizer for finding best solutions to goal programs, connected to find best values for said objective functions and constraints of said local party's goal program levelwise, and

a stay close processor for determining variable improvement directions from monitoring of received offers from said opponent and carrying out value perturbations in said directions,

said negotiator being operable to:

use said optimizer to produce a first offer for a first level, to advance from one level to another level only following acceptance by said parties of a unification offer regarding a previous level, and

use said stay close processor to produce a first offer for each subsequent level.

According to a fifteenth aspect of the present invention there is provided a platform for joint processing of goal programs to produce an outcome, the platform comprising:

a party goal program unit for formulation of at least one local goal program,

a unifier for determining common ground between said local goal program and at least one other goal program,

a negotiation necessity tester, associated with said unifier, for joint solving of said local and said other goal program to form a joint goal program comprising optimal solutions for each of said local and said other goal program, said negotiation necessity tester being set to determine whether there lies a single solution that includes both optimal solutions within said common ground, and if so, to indicate that no negotiation is necessary.

According to a sixteenth aspect of the present invention there is provided a resource negotiator for making successive offers for usage of a resource with at least one remote party based on a goal program of a local party, the goal program comprising a plurality of objectives, at least one of said objectives having a target value, an upper bound, a lower bound and at least one constraint, the resource negotiator comprising:

an input for receiving data from said remote party,

a minimizer for producing successively worsening minimizations of said goal program, and an offer formulator, associated with said minimizer, for formulating said minimizations into offers for resource usage for sending to said remote party.

According to a seventeenth aspect of the present invention there is provided a resource negotiator for negotiating for usage of a resource with a plurality of remote parties based on a goal program of a local party, the goal program comprising a plurality of objectives, at least one of said objectives having an upper bound, and a lower bound, the resource negotiator comprising:

an input for receiving data from said remote parties,

an objective maximizer for maximizing a value of said at least one objective, and

an offer acceptor, associated with said maximizer, for receiving offers from said remote parties, comparing said maximizing with said offers and for accepting one of said offers based on said maximizations.

According to an eighteenth aspect of the present invention there is provided a resource negotiator for negotiating for usage of a resource with a plurality of remote parties based on a goal program of a local party, the goal program comprising at least one objective assignable with at least one of an upper bound, and a lower bound, the resource negotiator comprising:

an active bid monitor for monitoring remote parties remaining in said negotiating,

a value increaser for successively increasing a value of said at least one predetermined objective, an offer acceptor, associated with said active bid monitor and with said value increaser, for ending said negotiation at a time at which only a predetermined number of remote parties remains active, and at a corresponding value of said at least one predetermined objective, said offer acceptor being operable to deem said negotiation successful if said corresponding value is within any assigned bounds, said predetermined number being related to a number of available resources.

According to a nineteenth aspect of the present invention there is provided a platform for performing ranking between database entries, each of said entries comprising a series of values arranged in fields, the platform comprising:

a trade-off unit for taking values from a plurality of fields of respective entries, defining a trade-off relationship therebetween such that for any given combination of values in said fields a single trade-off value is defined, and

a ranking unit for performing ranking amongst said entries in accordance with a respective single trade-off valuc.

Brief Description of the Drawings

For a better understanding of the invention and to show how the same may be carried into effect, reference will now be made, purely by way of example, to the accompanying drawings.

With specific reference now to the drawings in detail, it is stressed that the particulars shown are by way of example and for purposes of illustrative discussion of the preferred embodiments of the present invention only, and are presented in the cause of providing what is believed to be the most useful and readily understood description of the principles and conceptual aspects of the invention. In this regard, no attempt is made to show structural details of the invention in more detail than is necessary for a fundamental understanding of the invention, the description taken with the drawings making apparent to those skilled in the art how the several forms of the invention may be embodied in practice. In the accompanying drawings:

Figs. 1- 17 are simplified block diagrams which show various aspects of a platform according to the present invention,

Figs. 18-28 show various kinds of trade-offs and constraints for use with the platform of Figs 1-17, and

Figs. 29-50 illustrate uses of examples of the present invention.

Description of the Preferred Embodiments

The present embodiments describe a general purpose electronic negotiating platform that uses the concept of a goal program as the basis for allowing negotiations. The goal program can be formulated for users whatever their requirements and used in the conduct of the negotiations, thus freeing the platform from any particular format for the negotiations. Furthermore the goal program format does not require any particular type of goal such as price to form the centerpiece of the negotiation, thus extending the field of electronic negotiation beyond the business field.

Before explaining at least one embodiment of the invention in detail, it is to be understood that the invention is not limited in its application to the details of construction and the arrangement of the components set forth in the following description or illustrated in the drawings. The invention is applicable to other embodiments or of being practiced or carried out in various ways. Also, it is to be understood that the phraseology and terminology employed herein is for the purpose of description and should not be regarded as limiting.

Reference is now made to Fig. 1, which is a simplified diagram showing a platform for supporting negotiation between parties to achieve an outcome, operative in accordance with a first preferred embodiment of the present invention. The platform 10 comprises five basic units as detailed hereinbelow.

The first unit is a party goal program unit 12, which defines a respective party's goal programs in respect of the desired outcome. The goal program is a way of expressing a party's needs, constraints and desires regarding a particular outcome in quantitative form so that the party's position regarding the outcome can be compared in automatic manner with that of another party in order to ensure that the outcome takes on a mutually acceptable form. and comprises a plurality of objective functions which are expressions over variables (usually ones measuring deviation from objectives), partitioned into levels of importance, and constraints, and in particular goal constraints (which are 'soft' constraints representing a party's goals, or objectives) associated with respective objective functions. In fact the party goal program unit may physically be located with the respective party remote from the rest of the platform, or the platform may have a party goal program unit for each party involved or, as shown in Fig. 1, it may have one party goal program unit for a local party and simply expect a remote party to provide a goal program already formulated. As will also be discussed below, the remote party's goal program may not always be available. The negotiation may proceed in a mode known as "ignorant" in which some or all of the other party's goal program features are not known.

At the level of the party goal program unit 12 the possibilities for the outcome are expressed as a series of objective functions. The individual objective functions are then divided into levels of importance, for example primary, secondary and tertiary, and the objectives within each level are then preferably assigned importance weightings. Typically, the assignment of levels and of weightings to each objective function is carried out by means of a dialog box based interaction with the party.

Additionally, the platform 10 comprises a goal program unifier 14, which is connected to the one or more party goal program units 12 for receiving goal programs of the respective parties. The unifier carries out a task known as unification of the goal programs which involves determining whether two (or more) goal programs have a common field of interest from which a mutually compatible outcome is derivable. If there is no common field of interest then there is no point in making any further attempt to find a mutually acceptable outcome. Thus if a farmer wishes to use a field as a helipad, but is prepared to settle for a camp site, and the second party is the local noise abatement society which is totally opposed to aircraft noise but may be persuaded regarding tourist traffic provided that certain constraints are observed, then the unifier may identify the camp site use as the common field of interest.

Following the unifier 14 is a negotiator 16 which receives the goal programs of the respective parties, and carries out negotiations using the goal programs. Preferably, negotiations are carried out by considering the objective functions per se and considering them levelwise, that is to say by solving the objectives over a first level before moving on to a second level. Solving the goal programs involves a process of minimization of variables associated with the objective functions and will be explained in greater detail below. Each individual attempt at a solution between goal programs, or, in ignorant mode, between one goal program and what is known or can be assumed about another party, provides the content of an offer which is made to the parties.

The platform also comprises an output unit 18 formulating the content into an offer and sending it to the respective parties.

The platform further comprises a response receiver 20 for receiving from respective parties either counter offers or acceptances. The response receiver uses the counter offers as new input to the negotiator to arrive at a new offer. It will be appreciated that when both parties accept an offer, then the negotiation is complete.

The negotiator 16 preferably comprises a trade-off unit 22 which has a number of tasks involving the variables associated with the various objective functions. In certain cases it is able to identify certain objectives in a first party's goal program and to offer a conditional weakening of that objective in return for strengthening of other objectives within the same level. In one embodiment, the other objectives may be linked with the first so that the trade-off is meaningful.

Another kind of trade-off that the trade-off unit may involve itself in concerns identifying certain objectives at a given level in a first party's goal program which the first party is prepared to have weakened in return for concomitant weakening of certain of the other party's objectives. The trade-off may involve weakening of other objectives within the same level of the other party, if known. In ignorant mode the importance attached by the other party to his objectives is not known.

As mentioned above, the party goal program unit 12 preferably expresses objective functions within the goal program in a weighted hierarchy according to a respective associated level of importance, typically provided by the user but in the absence of user input, default values can be used. Each constraint is preferably expressed in terms of a constraint variable. Between them, the goal program is formed into a series of expressions, one per level, suitable for minimization within the negotiator.

Data from the user input is preferably used to apply coefficients to the constraint variables.

Typically, the party goal program unit 12 applies user input to provide different values of coefficients to the constraint variables for deviations of a corresponding objective in respective positive and negative directions. That is to say a given variable has a preferred region or target value which the party wants to achieve. He objects to the variable being outside that preferred region or away from that target value, and his objection is expressed in terms of a coefficient. The variable can take values above or below the preferred region or target value and the user may apply different coefficients for the different directions, expressing different levels of dislike for the respective directions. Thus, for example, in scheduling a train service, a railway official may have an ideal time range for the journey to take. Deviation above the range is allowable to a limited extent but beyond that to be discouraged strongly because of unsafe speeds. Deviation below the range is discouraged as it is unpopular with passengers, but no question of public safety arises. Using a negative coefficient indicates a desired deviation towards a direction, also known as a bonus.

In general, a certain class of objectives may be covered by continuous variables, and constraints are applied as coefficients to parts of their ranges. The variables may be categorized according to the arrangements of coefficients in various ways, for example a strong one sided objective is an objective with a target value or region of preference and a strong weighting on one side of that region or value. A weak one sided objective is the same but with a weaker weighting to the side of that region. A complex single sided objective has two (or more) different weightings making up two different parts of an otherwise continuous non-preferred region. A simple two sided objective has single weighting coefficients on either side of a centrally included target value or preferred region. A complex two sided objective has two (or more) different weightings on both sides of a centrally included target value or preferred region. A simple first side-complex second side objective has different weightings on one side and a single weighting on the second side of a centrally included target value or preferred region. A simple two-sided objective with an indifferent range is specifically a simple two sided objective in which a centrally included region is entirely flat, that is to say has no preferences associated with it at all, likewise a complex two sided objective with an indifferent range, and a simple first side-complex second side simple objective with an indifferent range.

The above applies to continuous variables. However, as will be described in greater detail hereinbelow, the platform may also take into account objectives that are described by a series of discrete values. In such a case, the party goal program unit 12 applies user input to formulate weightings for the different discrete values. Thus it is possible to express a preference between the discrete values. As will be discussed in more detail below, the discrete variables are transformed into a continuous plane for processing and then transformed back into discrete form for offer formation. The discrete variable allows the goal program to express objective functions involving discontinuous values. For example a proposed land use objective may be expressed as a series of discrete values, say, 1) heliport, 2) camping site, 3) industrial building, 4) commercial building, 5) domestic building, 6) agricultural use. It will be appreciated that a continuous variable would be of little use in such circumstances.

Notwithstanding the presence of discrete variables, the main workhorse of the goal program is likely to be the continuous variable, and the party goal program unit may generally apply user input to determine whether the continuous variable is to be maximized or to be minimized, once the above-discussed constraints have been applied. For example an important variable in many circumstances is time, time to completion or the like, and the direction of desirable movement of time depends on who the party is. A passenger wants a shorter time. A transportation provider would rather have the flexibility which a looser determination of the time would give.

The negotiator 16 preferably deals with objectives as arranged in a hierarchy, both of levels and of the objectives within the level and preferably carries out an activity of minimization over the goal program levels. Minimization typically involves summing objective functions associated with individual goals (i.e., objectives) and associated constraints. The minimization is preferably carried out level by level in the hierarchy, which is to say that first one level is solved for all its objectives and then the negotiator moves on to the next level. Preferences within the levels are used as part of the weightings within the objective function. This point is made in terms of mathematical equations hereinbelow.

The negotiator may carry out minimization per objective function or constraint variable, and may set a maximum bound per deviation. Such a maximum bound is used to ensure that the system does not arbitrarily select a variable and stretch it beyond realistic expectation. Such a selection may succeed in balancing the opponent party mathematically but produces unrealistic offers.

Preferably, the negotiator carries out minimization for an expression comprising higher priority constraint variables and then iteratively adds further constraint variables to the expression for further minimization stages.

Reference is now made to Fig. 2, which is a simplified block diagram showing in greater detail the goal program unit 12 of Fig. 1. The goal program unit 12 preferably comprises an input unit 24, a default value generator 26 and a prioritizer 28. Preferably, the input unit 24 is configured to receive data from a user interface, for example through a dialog box as discussed. As an alternative, the input is configured to receive data from a software agent. In a particularly preferred embodiment data can be accepted, by the input unit 24, from either kind of source.

As mentioned above, goal programs can be completed with default values, and for this purpose, the party input unit 12 identifies parameter data missing from an input and further comprises a default value generator 26 which may generate a default value for the missing parameter according to a predefined criterion. Typically the generator 26 may use a default register of values for expected parameters.

The party input unit preferably obtains lower and upper bounds for at least some of said objective functions. Again these bounds may be obtained directly from the user input, adapted from the user inputs or default values may be used. The use of upper and lower bounds to a parameter ensures that parts of the range expressed by the parameter may be expressed as a proportion of the whole, thereby rendering different ranges comparable. The goal program unit 12 is thereby enabled to use the upper and lower bounds to express deviations of a objective function from a target value relatively, thereby to render the deviations subject to comparison by the unifier 16 or by the negotiator 18.

The party input unit 12 preferably obtains an objective function interval, and a value for a penalty for deviating from said interval, as described above. Subsequently, the unifier may define a working interval between two objective functions as an intersection between the respective intervals.

The unifier preferably determines that a target value of one of the objective functions is outside the working interval, and modifies the target value to approach a closest boundary of the working interval.

The unifier is operable to apportion the penalty in accordance with the target value modification.

In the above modification, the intersection may be a point.

The party goal program unit 12 preferably is able to determine that an intersection is small to the satisfaction of respective parties. When the intersection is recognized as small, the negotiator selects a point within the intersection which is a midpoint between respective target values.

Returning to the issue of bounds and the use of bounds to define any kind of interval in respect of a variable allows the negotiator to measure deviations within the interval as a fraction of a total size of the interval.

The party goal program unit 12 preferably uses data from the user to obtain importance values for deviations from the target. The importance values can then be used by the negotiator as multipliers to scale deviations from the target values.

Preferably, the negotiator 18 identifies intersections that are small and distant from a target value compared to one of said objective functions and large and inclusive of a target value compared to another of the objective functions. In such circumstances, the negotiator attempts to set an effective target at the closest intersection boundary and then applies a transformed deviation. That is to say, since, from the point of view of the more distant objective, an artificial target value is being used, the deviations from the artificial target cease to reflect the actual cost of the deviation from the point of view of the respective party. Thus a transformed deviation is calculated from the arithmetic deviation when computed relative to the effective target and then added to the difference between the old target and the effective target, then to produce a result which is divided by the old target. The calculation is shown mathematically in the examples section below.

The party input unit preferably permits a party by answering questions at the user interface, to define objective functions according to any one of the categories discussed above. Thus for example the user may define a single dimension indifference interval objective and may associate the objective with a range of indifference having an upper bound and a lower bound, a first weighting value for deviations below the lower bound, a second weighting value for deviations above the upper bound and a relative importance for the objective as a whole.

Reference is now made to Figure 2. The unifier 16 may then use the range of indifference, the weightings and the relative importance to unify the objective with opponents' objectives to determine whether or not there is a common area of interest.

The prioritizer 28 allows a respective party to define an association of a respective objective function with other objectives. The association can then be used to choose which levels the objectives should be placed, whether on the same levels or on different, say succeeding levels. The user may indicate that give on one of the linked objectives should correspond to take on the other. Likewise the user input may be used to establish a priority amongst objectives within a single level.

The input may also permit a party to define a two dimensional trade-off objective by entering two two-dimensional points. The party goal program unit 12 preferably defines a trade-off line between the two points, as for the one dimensional version. The principle may be extended to three and higher dimensions.

The multi-dimensional trade-off includes a similar facility for defining weights for deviation from the trade-off line.

The input unit preferably allows a party to define a relative importance for the two (or higher) dimensional trade-off objective.

Again, as with the single dimensional trade-offs, the input unit permits a party to associate the two dimensional trade-off objective with other objectives at a desired level.

The input unit preferably allows a party to define a single dimension two-point objective, in other words an objective having more than one target value. The objective may thus be associated with an upper point of preference, and a lower point of preference. The two points of preference divide the objective into three regions, a first region above the upper point of preference, a second region included between the points of preference and a third region which is below the second or lower point of preference. Separate weighting values may be attached to these regions, thus a first weighting value for deviations below the lower point of preference, and a second weighting value for deviations above the upper point of preference. Generally the included region is a region of indifference but if weightings are to be attached thereto, then generally two are required. The goal program unit is preferably able to provide weightings to the included region by assigning a first weighting value below said upper point of preference and a second weighting value above said lower point of preference. Then an overall weighting is defined within the included region as the minimum of the two weightings. The application of the two weightings will be explained both graphically and mathematically in the examples section below.

In addition, the input unit preferably allows a party to define a relative importance for the single dimensional two point objective, and generally applies to the two point objective any other of the features applied to any of the other types of objective discussed hereinabove.

The input unit preferably permits a party to associate the single dimensional two point objective in levels with other objectives. As before, the relative importance applied to the two point objective is usable by the negotiator to establish a priority between objectives within any given level.

The input unit preferably permits a user to define a piecewise linear two-dimensional objective by entering at least three two-dimensional points. The goal program unit or the negotiator is then able to define a trade-off line between the three points.

The negotiator applies penalty values to points away from the trade-off line in accordance with their distances from the line.

The party input unit preferably permits a user to define a first deviation weight for deviating in a first direction from the trade-off line and a second deviation weight for deviating in a second direction therefrom.

The party input unit preferably permits parties to define objectives comprising pairwise trade-offs having at least two points and a trade-off therebetween, and to associate each of the trade-offs with a slope.

The party goal program unit preferably prevents inconsistent trade-offs to be defined within the platform by preventing the input unit from accepting more than one trade-off from referring, directly or transitively, to any already specified trade-off, or less stringently, that defines a different trade-off between variables than one already specified. Alternatively, the goal program unit may simply warn users of inconsistent trade-offs by outputting a warning whenever a trade-off being entered contradicts, or more stringently, potentially contradicts, with a previously defined trade-off.

Reference is now made to Fig. 3, which is a simplified diagram showing the trade-off unit 22 of Fig. 1 in greater detail. The trade-off unit comprises a disjunctive constraint processor 30, whose operation will be described below, a trade-off line evaluator 32 and a scaling unit 34. The trade-off line evaluator 32 preferably carries out the trade-offs described hereinabove and the scaling unit 34 preferably applies scaling based on upper and lower bounds to make different parameters comparable.

Preferably, the party input unit permits parties to define disjunctive constraints in respect of objectives, and to this end the disjunctive constraint processor 30 translates a disjunctive expression into at least one linear conjunctive expression. The unifier and the negotiator are then both able to use the translated expression for their respective purposes.

The disjunctive expression may typically include a series of relationships, and the individual relationships may include equality relationships.

Preferably, the disjunctive constraint processor carries out translation by expressing any one of the equality relationships as the union of two corresponding inequalities that meet at a point of equality of the equality relationship. This point and the necessity for it is discussed at greater length in the examples section below.

The disjunctive constraint processor allows binary variables to be defined in respect of the relationships, for setting wherever said relationships are satisfied, and wherein said negotiator is operable to sum said variables to determine a satisfaction level for said objective.

The party goal program unit may set a requirement of a minimum number of satisfied relationships for use by said negotiator.

The party input unit preferably permits each party to define weighting values for a discrete variable which is predefined per outcome, so that the definition can then be used setting up all of the goal programs for the outcome. The negotiator is then able to carry out negotiation between the goal programs by considering the weighting values and arrives at one of the values to make as an offer.

The party goal program unit preferably uses weightings each of the different discrete values that the variable is able to take. The discrete values can then be arranged as a weighted hierarchy. Subsequently, the hierarchy can be used by the negotiator to choose between the various discrete values for inclusion in an offer.

The party goal program unit preferably uses the values and corresponding weightings to build summation functions. The summation functions will be discussed in greater detail in the examples section below and enable expression of the discrete variable in quantitative manner.

The negotiator may involve the discrete variable in offer formation by a process of setting one of the discrete values to one and the remainder to zero, and then calculating the summation function and using it to contribute to an overall fulfillment level of each goal. The process can be repeated with different values to arrive at different attempts at an offer, thereby to find a value which maximizes the fulfillment level.

As mentioned above, time information can often be significant in outcomes, and the platform therefore requires a way of measuring time that is directly applicable to the trade-off mathematics that has been described above. The party goal program unit therefore preferably represents date information as accumulated minutes from a threshold starting date, so that it takes on the characteristics of a continuous variable. The goal program unit is further operable to modify the dates relative to upper and lower bounds entered via the party input unit, in the same way as described for any other kind of variable.

The input unit further allows input of variables in association with given objective functions and allows setting of a linkage between the variables. The linkage preferably defines a trade-off relationship of deviations with respect to respective target values. The negotiator uses the series of variables, including the trade-off path, to carry out negotiation involving other objectives to arrive at formation of an offer.

Preferably, the party goal program unit expresses the second variable as a function of the first variable in order to represent the linkage. The negotiator then uses the linkage to represent deviations from second variable target values as deviations from the target value of the first variable.

The goal program unit may express the trade-off as separate deviation variables in respect of the first variable and in respect of the second variable. The separate deviation variables are preferably orthogonal.

The input unit preferably permits an association of a relative importance level to the trade-off, as for the previous examples.

In a possible embodiment, the party goal program unit calculates a relative importance level for the trade-off as an average of respective relative importance levels of the first and second variables.

Reference is now made to Fig. 4 which is a simplified block diagram showing the unifier of Fig. 1 in greater detail. The unifier 14 comprises a goal program generalizer 36 to form a generalization of received goal programs for use in subsequent processing. The generalized goal program, otherwise referred to as a joint goal program, combines the constraints of both the local and opponent goal programs. Following the goal program generalizer 36 is a negotiation necessity tester 38, whose purpose is to determine whether any advantage can come to either party from negotiating, that is to say, it determines from the objective functions and constraints, whether there already is a single unambiguous solution of maximal advantage to both or each party. If there is then such is presented as the offer and there is no point entering into negotiations.

Preferably, the goal program unit translates the input received from the input unit into objective functions, and into constraints on the objective functions within the goal program.

Reference is now made to Fig. 5, which is a simplified diagram showing in greater detail a first part of the negotiator 16. The negotiator 16 is shown with trade-off unit 22 as discussed above, and in addition has an optimizer 40 whose task is to find best values for the objective functions and constraints. The optimizer 40 then uses the best values to obtain a best solution for the local goal program, bearing in mind constraints of the opponent party, for output as a first offer. The optimizer 40 then iteratively produces further solutions, each possibly respectively worse from the point of view of the local party goal program until an offer is accepted. In the case where the opponent's goal program is known, the optimizer may produce improvements in the opponent's goal program with possible worsening of the local goal program.

The negotiator preferably further includes a percentage reducer 42 which is connected to the optimizer 40, and which serves to take offers, worsening them by a predetermined percentage, and therefrom to produce the additional offers.

In a preferred embodiment, the percentage reducer, which works level by level, can be set to take each of the variables within a level in turn according to orders of weighting within the level, until a solution is accepted.

The negotiator 40 preferably additionally comprises a worst case calculator 44 for determining a worst case level for individual objective functions and constraints, thereby to obtain a worst acceptable offer, the minimum acceptable to the local party. As will be discussed in more detail below, having a worst case offer acts as a boundary for a negotiation space against which movement between offers can be measured. For example it becomes possible to decide that each succeeding offer may approach the worst case boundary by a certain percentage of the space remaining.

One or more of the goal program variables may be arranged as pairwise bounded variables, that is to say variables covering alternative positions so that in any given offer one of them will be set to a value within certain bounds, and the other to zero. The negotiator 16 preferably additionally comprises an arbitrary case calculator 46 for taking one of each pair of variables, setting it to an arbitrary value within its respective bounds, taking the other of said pair of variables and setting it to zero, and using the various arbitrary settings to arrive at different solutions.

The negotiator further comprises an average case calculator 48, also connected to the optimizer 40 and to the worst case calculator 44. The average case calculator takes the best solution and the worst solution, and the variable values that correspond to them, associates corresponding best and worst values together and finds an average. The goal program is then constrained towards the averages for each variable and an average solution is produced.

The average case calculator preferably makes use of the arrangement of variables into levels of importance to process the variables level by level. Such level by level processing may be used to produce a series of iterative solutions to serve as offers.

In the above, constraining towards the average produces deviations from target values, as the average and the target values are not necessarily the same. It is therefore desirable to minimize the deviations, and this minimization is carried out level by level, with weightings being attached to the deviations at different levels so that the deviations at the higher levels are weighted more strongly and therefore can be more affected by the minimization.

The negotiator further includes a solution unit 50 for applying preferences to different goal program solutions produced using the optimizer 40. The preferences are used in producing offers. The solution unit is shown in greater detail in Fig. 6, to which reference is now made. The solution unit comprises a solution sorter 52 which carries out a comparison between goal program solutions by calculating the goal program for each one of a series of solutions (set of values provided using the optimizer) and then ranking the solutions. The negotiator 16 then uses the ranking to apply preferences to different solutions.

The negotiator 16 additionally includes a thresholder 54 which is connected to the solution sorter, and which applies a threshold to the evaluations. Any solution not meeting the threshold can be rejected as desired.

Preferably, the solution sorter further comprises a solution completer 53 for applying best values to unspecified variables in incomplete solutions, thereby to allow said goal program to be evaluated for said incomplete solutions. The reason some of the solutions may be incomplete may be due to incomplete data received from the respective party. Even if a default value generator is used as described above, the situation of an incomplete goal program may still arise for a number of reasons. For example, the situation behind the goal program may not be sufficiently well defined for the default value generator to help, or the lack of completeness may be due to the absence of something more complex than the simple lack of a target value, variable boundaries or weighting values. For example, the actual behavior of a variable as its values change may be absent.

Typically, the solution sorter 52 is set to find the best solution from the series by identifying the highest ranked solution and discarding the remaining solutions. In a preferred embodiment, the solution sorter 52 comprises a memory 54, which holds a certain number of solutions. A comparator 56 compares any new solution with each solution in the memory 54, and a control unit 58 adds the new solution to the memory 54 if its evaluation is larger than any solution currently in the memory. In such a case the lowest ranked solution currently held is typically discarded.

Preferably, the input unit is able to accept user defined output values, that is variables in which the particular party desires a single output and is not interested in any negotiation in respect thereof. The goal program unit sets the output values as single value constraints and flags the constraints as unchangeable for subsequent processing. Thus for example a fisheries authority may use the system to negotiate fishing quotas for a particular type of fish and may be prepared to accept give and take on a wide variety of issues but is not prepared for any negotiation on a minimum net size. In such a case the minimum net size is selected as a user defined output variable.

User defined output variables acting on the goal program as a single value constraint can cause problems in finding any feasible solutions for the goal program. Preferably, the data input unit is therefore set to output an error indicator if one or more single value constraints render the goal program insoluble. In other cases the goal program may proceed to the unifier which may fail to find a common region of interest. Thus the user defined output variable is not something to be encouraged, but for the platform to be realistic it cannot be excluded.

As discussed above in respect of Fig. 5, the optimizer 40 finds best solutions to goal programs. The solutions give best values for the various objective functions and constraints of the local party's goal program by proceeding in a level by level process. The worst case calculator 44 finds worst solutions for the goal programs, and itself finds the worst values for the objective functions and constraints level by level. The worst acceptable case can be calculated for the local party or for the opponent if his goal program is known. Either way the worst case is used in the same way, as a way of determining appropriate distances to move between offers, either towards the local party's worst case or away from an opponent's worst case.

Another way of using the worst case calculator is illustrated in Fig. 7, to which reference is now made. The negotiator 40 preferably uses the optimizer 40 and the worst case calculator in succession level by level between the goal programs to produce successive value sets or solutions for evaluation. From these successive best – worst sets, offers are made for the current level only. The platform only advances to a succeeding level once an offer is accepted at the previous level.

The negotiator 40 preferably comprises a constraint updater 60 for updating constraints upon advance from one level to another level in accordance with the acceptance of the previous level. The accepted objective function values of the previous level are taken as fixed constraints for the succeeding levels as far as is possible.

Reference is now made to Fig. 8 which is a simplified block diagram showing further features of the negotiator 16. In Fig. 8 the features involved in solving the goal program for optimum and worst cases etc. and subsequently selecting the solutions for formation into an offer are included as a single block 62 referred to as a goal program (GP) evaluator. The goal program evaluator 62 is connected to an offer unit 64, which takes the selected solutions, generally a set of values, and formulates them into a meaningful offer for output to the parties. The local party may require to approve the offer before it is sent to the opponent and the opponent then may approve the offer or reject the offer or make a counter offer. Either way, the opponent's input is accepted as feedback which is sent to an offer improver 66. The offer improver 66 preferably makes a change in a selected one of the variables to bring about an improvement in the evaluation of the opponent's goal program if known. If the opponent's goal program is not known then generally the best strategy is to worsen the evaluation of the local goal program, on the assumption that a local worsening equals an improvement for the opponent. A further strategy is to find a value that has been improved from the local party's point of view in an opponent's counter offer. Improvement by the opponent suggests that the opponent believes he is paying a price and thus suggests that the corresponding variable is a matter of importance. Another strategy is to do the exact opposite, that is to find a value that the opponent has not budged on in his last counter offer. Failure to budge may indicate that the value is important, and if the local party is prepared to be flexible then progress may be made.

An important question in offer improvements is what level of change to incorporate in successive offers. Negotiations can break down if either party perceives that progress is not being made. Likewise a party may be wary of showing weakness if it concedes too fast, and in any case is likely to get a raw deal if it makes concessions at a faster rate than the opponent. In one preferred embodiment a change between successive offers is a proportion of the difference between the previous offer and a best possible evaluation made for the opponent. Likewise it could be a proportion of the difference between the previous offer and the worst acceptable value for the local party, as has been referred to above.

In the case where the opponent's goal program is known and the offer improver 66 is being used to find improvements in the opponent's goal program, a problem arises that, because two goal programs are not symmetric, a modest improvement in the opponent's goal program may actually correspond to a drastic worsening of the local goal program. In order to mitigate against such an eventuality, the offer improver 66 may calculate what is known as a protection value. The protection value is typically used to set a limit to an allowable reduction in the evaluation of the local party's goal program over a single offer cycle as a consequence of improvements to the opponent's goal program evaluation.

Typically, the protection value is a proportion of the difference between a worst case evaluation of the local party's goal program and the previous evaluation.

In one strategy for successively improving offers, generally used when the opponent's goal program is not known, the offer improver 66 takes goal program values of a previous local party offer and one value in turn from a previous opponent offer. The opponent value is tested against local constraints, and, if it fits within the constraints, then it is substituted into the previous local party offer. The previous offer with the newly substituted opponent value is then submitted as an improved offer.

An oft-used tactic in negotiations is the use of time between offers. Experienced negotiators use time as a tool to intimidate opponents or to send a message to an opponent regarding the direction of negotiations or to give the impression that important and weighty factors have to be taken into account in considering a previous offer. The platform provides such a facility for negotiators via offer timer 68, which is programmable to set different kinds of delays between offers. It may set successively increasing delays, or the delays may change per changes in level, or a magnitude of the delay may be based on a relative change between succeeding opponent offers, or it may simply use user determined delays.

The use of offer timer 68 is discussed in detail below in the examples section.

Reference is now made to Fig. 9, which is a simplified block diagram showing a variation of the embodiment of Fig. 7. Parts that are the same as those in previous figures are given the same reference numerals and are not referred to again except as necessary for an understanding of the present embodiment. In the embodiment of Fig. 9, a stay close processor 70 determines variable improvement directions from monitoring of received offers from the opponent and carries out value perturbations in the directions that it finds that the opponent has been using.

The use of a stay close processor essentially mirrors the other party's moves and therefore is an appropriate strategy if the opponent's goal program is not known. Of course it cannot be used to make an initial offer since opponent movement directions are not yet available. A way of using a stay close processor in conjunction with level by level negotiating is as follows:

Firstly the optimizer is used to produce a first offer for a first level, since no directional data is available from the opponent. Subsequent offers are made via the stay close processor.

As before, advances from one level to another level are made only following acceptance by the parties of an offer regarding a previous level.

The stay close processor can be used to produce a first offer for each subsequent level.

As with Fig.7, the constraint updater sets already agreed objective function values from previous levels as constraints for the later levels.

Reference is now made to Fig. 10, which is a simplified block diagram showing a variation of the embodiment of Fig. 8. Parts that are the same as those in previous figures are given the same reference numerals and are not referred to again except as necessary for an understanding of the present embodiment. In the embodiment of Fig. 10, a gap value determiner 72 is used to find a gap that can be used in subsequent offer improvement. A value improver 74 is connected to the gap value determiner 72, for inserting a proportion of the gap as a constraint into the goal program. Subsequent to and consequent on the evaluation of the goal program with the new constraint, the stay close processor 70, which is connected to the value improver 74, applies the gap proportion in the appropriate direction to provide an improved offer.

The stay close processor 70 monitors two successive opponent offers for value changes therebetween, and preferably assigns weights to the various changing variables. The weight is subsequently used in providing an improved offer. The stay close processor 70 preferably selects the magnitude of the weights in accordance with the amount of change applied by the opponent, so that the use of the weights provides a strategy for mirroring the opponent's activity in offer improvement.

Returning to the gap processor 72 and the gap used in each offer may be a fixed proportion of the gap for the entirety of the negotiation. That gap may be a difference between a best value and a worst value of the corresponding variable. Alternatively, the gap may change between successive rounds. For example the gap may be the difference between the last local proposal and the last opponent proposal.

Referring back to the negotiation necessity tester 38, which is part of the unifier, the negotiation necessity tester is preferably set to determine whether there lies a single solution that includes both optimal solutions within the common ground of the two parties. If such a single solution exists then passing of the goal programs to the negotiator is preferably inhibited since there is nothing to be gained by negotiation.

Reference is now made to Fig. 11, which is a simplified block diagram showing further details of the negotiator 16. Not always does negotiation succeed. Certain levels of the negotiations may prove intractable to the parties. The platform therefore preferably includes a mediation unit 78, which may be called by agreement between both (or all) parties upon failure to negotiate a given level during the negotiations. The mediation unit 78 retains agreed objectives of previously solved levels and applies a summation formula to solve a current level.

As discussed below in the examples section, a typical summation formula is

$$\left\{\frac{\sum w_k^+ \cdot \delta_k^+ + \sum w_k^- \cdot \delta_k^-}{\sum w_k^+ + \sum w_k^-}\right\} + \left\{\frac{\sum v_k^+ \cdot \gamma_k^+ + \sum v_k^- \cdot \gamma_k^-}{\sum v_k^+ + \sum v_k^-}\right\}$$

wherein k is a number of a current level, objective, + and – represent respective sides of a target value, v is an importance level, w is a weighting and γ is a goal program constraint value. As an alternative, the summation formula may be a median solution.

The mediation unit 78 may use a summation formula to provide a median solution between entire goal programs if the negotiation is not a level by level negotiation or if the negotiations are interminably stuck and level by level mediation is believed to be inadequate.

Reference is now made to Fig. 12, which is a simplified block diagram showing further details of the negotiator 16. As discussed above, each goal program is expressed as a series of objective functions involving decision variables each having an upper bound, a lower bound, a target value or values or an indifference interval and one or more constraints. The platform preferably comprises a form offer unit for providing a form offer to the parties. The form offer unit provides a solution without negotiation but is not the same as the mediation unit. It can be used as a mediation unit in the event that negotiations reach an impasse or it can be used directly to make an offer to the parties in the event that the parties do not wish to negotiate. The form offer unit works by assigning to each of the goal programs a weighting such that the sum of the weightings is unity over each variable over the goal programs. It then calculates the offer by minimizing relative deviations for each variable over the goal programs for the assigned weightings. The form offer unit is described in greater detail in the examples section.

The form offer unit preferably includes a discrete variable unit which can transform values of a discrete variable into a continuous domain, to carry out minimization in light of goal program objectives of the two parties. As discussed above, the discrete variables are evaluated in a continuous domain, and the minimization results are translated or transformed back to discrete values for presentation in the form offer.

The negotiator 16 may be used to provide a value for just one objective by comparing the single objective, from a local goal program, against the entirety of an opponent goal program. Such a procedure is useful in an auction type of situation where a single party offers something to a plurality of opponents and selects a winner. In general an auction situation is one in which one party acts against several opponents to choose a winner. The single party may be offering something or inviting tenders for supply of a product or service. Particularly in the latter case the auctioneer may have a relatively complex goal program, although typically most of the goal program consists of fixed values with only one or a small number of variables open for negotiation. A range of types of auction are discussed in detail in the examples section below, including the English auction, second price auction, and reverse auction or tendering.

Reference is now made to Fig. 13, which is a simplified block diagram showing further details of the negotiator 16 for the specific application of matching a goal program of one party having a catalog of items, services, packages etc. and a second party who has a range of requirements that he hopes will be met from within the catalog. The figure shows an item catalog 82 for storing representations of the items for offer in the catalog. The items are represented in terms of values for variables in the objective functions of the corresponding goal programs, and in general, the negotiator deals with the items by solving goal programs as described above and then using the solution values to identify the nearest corresponding item in the catalog. The negotiator 16 includes an item manager 84 which recursively determines, during the negotiations which of the catalog items are currently within the scope of negotiations, although it is stressed that negotiations are not at this stage carried out on the basis of individual catalog items but rather on the basis of goal program objectives and values as before, with matching to closest items carried out only later. As will be explained in more detail in the examples section below, such an approach reduces the computational complexity involved in finding a solution.

A first round manager 86 is connected to the item manager 84, and manages the level by level goal program negotiation in respect of the catalog items by controlling the item manager to successively reduce the number of catalog items within the scope of the current negotiations to a predetermined threshold number of items.

A second round manager 88 is also connected to the item manager 84, and manages level by level program negotiation to produce successive offers. An item associator 90, which is connected both to the second round manager and to the item manager 84, expresses the successive offers in terms of items that remain within the scope of the negotiations. That is to say that the party is given a list of one or more items from the catalog, that answer to his preferences, to choose from.

In a preferred embodiment, the item manager 84 uses the goal program of the party selecting the item, rather than of the owner of the catalog, to measure distance from the current scope. In a further preferred alternative, the item manager can measure distance from the current scope in terms of a joint goal program.

Notwithstanding which goal program is used later on, the item manager 84 may begin by measuring distance in terms of a local goal program, to order the items within the catalog, and then iteratively to remove the most distant items.

In certain cases an objective of one of the parties may be compatibility with a second item. For example a party may wish to search a catalog for a trailer compatible with a given car, or alternatively a party may wish to find a car that accords with a certain goal program, and which can also take a certain trailer. In such a case, compatibility is simply entered as one of the objectives in the goal program.

In a preferred embodiment of the invention, one or more of the objectives in the goal program can be a dynamic variable. That is to say a particular objective may relate to some kind of changing situation. For example a goal program for optimizing sea transportation routes may relate to the weather.

As well as the objectives themselves, some of the constraints may be associated with dynamically changing variables.

The unifier has been discussed above as providing the facility of finding a common area of interest in which to proceed with negotiations. Such a feature provides a certain amount of screening in that completely useless negotiations are not entered into. It is also possible to carry out a stage of prescreening based on parties business intentions, perhaps on the size of the common area of interest, in order to weed out parties less likely to produce a good outcome. Such a screening stage is preferably carried out when there a large number of parties for potential negotiation.

Reference is now made to Fig. 14, which is a simplified block diagram showing a resource negotiator 100 for making successive offers for usage of a resource with at least one remote party based on a goal program of a local party. The platform is the same as that described before except that in Fig. 14 the creation, unification and solution of goal programs is assumed and the figure relates to the use of the solutions in formulating offers. The goal program comprises objectives as before, the typical objective having a target value, an upper bound, a lower bound and at least one constraint. The resource negotiator firstly comprises an input 102 for receiving data from the remote party, a minimizer 104 for producing successively worsening minimizations of the goal program, and an offer formulator 106, which is connected to the minimizer 104, and which is able to formulate minimizations into offers for resource usage. The offers may then be sent to the remote party.

Data from the remote party is a goal program like any other, typically comprising a plurality of objectives, the objectives generally having a target value, an upper bound, a lower bound and one or more constraints. The resource negotiator additionally includes a maximizer 108 for producing successively improving maximizations of the remote party goal program, for use together with the minimizations for formulating the offers.

The offer formulator may typically comprise a behavior synthesizer 110 for governing offer formulation in accordance with a selectable behavior profile, for example, aggressive, co-operative etc.

The behavior profile may include an opponent offer feedback feature (F/b) 112 which uses the extent to which an opponent's offers provide improvements, in order to choose how much to improve successive offers.

The feedback feature may use data from the opponent's offers to set time intervals for successive offer outputs. Thus small changes made by the opponent may be greeted by long waits or any other strategy deemed appropriate by the user may be set.

It is often necessary or desirable to break off negotiations at some stage, as a tactic or simply because of lack of progress. The behavior profile may therefore include a negotiation refusal condition unit 114 for breaking off negotiations when a predetermined condition is achieved. The condition set can be as simple or complex as desired, examples include a condition defining a predetermined number of opponent offers that fail a preset improvement threshold. Another possible refusal condition is a preset time interval during which the negotiation fails to reach a preset improvement threshold.

The maximizer may be associated with an offer acceptor which simply compares a current offer with a maximum calculation, and any offer that is sufficiently close is accepted.

Reference is now made to Fig. 15, which is a simplified block diagram showing an embodiment of the resource negotiator specifically for an auction type situation (i.e., implementing an auctioneer), that is to say for a one to many negotiation situation, and in particular where just a single objective is being negotiated. The resource negotiator 120 sends offers to and receives data from remote parties 122.1..122.n. An active bid monitor 124 monitors the remote parties as they remain in the negotiations and follows them as they drop out.

A value increaser 126 successively increases the value of the objective as the negotiation proceeds. An offer acceptor 128, is connected to both the active bid monitor and the value increaser, and ends the negotiation when only a preset number, typically one, of remote parties remains active, and at the value of the objective at the time. The offer acceptor 128 may check that the final value is within some kind of bound of acceptability before accepting the offer.

A bound assigner 130 may be provided to calculate such a bound according to other objectives of a local goal program.

Often, particularly in a reverse auction situation, it is desirable to send at least some of the details of the local goal program objectives to the remote parties. For this purpose there is provided a goal program output unit 132 which sends out selected local goal program objectives. The details can be used to enable the remote parties to evaluate potential offers in light thereof.

In certain cases, the method of successively raising a variable may lead to a situation in which at one level there are two many active parties and at the following level there are too few. In such a case it is useful to be able to find an intermediate value and the value increaser preferably includes a facility for this.

The active bid monitor 124 optionally comprises an output unit 134 for revealing to the remote parties how many other remote parties currently remain in the negotiations. The information may be used by the remote parties in various ways to support a negotiation strategy. Optionally, the output unit may reveal identities of the remaining parties.

A feature of one-to-many type negotiations is the need to decide when to drop out of the negotiations. Reference is now made to Fig. 16, which is a simplified diagram showing a drop out decision unit 140 for use by the remote parties in a one-to-many negotiation.

The unit comprises a current offer evaluator 142 for expressing a current value in terms of the remote party's own goal program. There is no point continuing if the current value exceeds the constraints of the local goal program.

An active bid unit 144 stores the number (and identities if available) of remote parties remaining in the negotiations.

A drop out function 146 calculates a drop out probability as a function of the current value and the number of remote parties remaining in the negotiating, and for that matter any other available data that the user may choose to insert into the function. A decision maker 148 uses the drop out probability to decide whether to leave the negotiations.

Reference is now made to Fig. 17, which is a simplified block diagram showing a platform for performing ranking between database entries. Preferably, each of the entries comprises a series of values arranged in fields, and goal programs are used to rank the entries, not in terms of a single field as is generally carried out at present, but rather on the basis of the goal program as a whole.

The platform comprises a trade-off unit 152 for taking values from the various fields, defining a trade-off relationship therebetween, and a ranking unit 154 that carries out ranking amongst the entries in accordance with a single trade-off value.

The trade-off unit 152 may take the field values in twos, so that each trade-off value is a trade-off between two fields.

In the trade-off itself, a reduction in one variables may be traded for a proportional increase in a second variable, or any other suitable trade-off method may be used.

In another preferred embodiment, the trade-off unit may take values in groups of three or more.

The trade-off unit may take the field values in trade-off groups, and may for example compile a separate trade-off statement for each group. The trade-off statement may include a deviation over the trade-off in the group from a target value. The trade-off statement may comprise compatible variables. Overall ranking is then achieved by the ranking unit by using an average of the deviations for each trade-off statement. Thus, once again, ranking itself is achieved using a single value.

The trade-off relationship may comprise an inequality between corresponding values of successive entries.

The trade-off unit 152 may also include a weight assigner 156 for assigning weights to fields. The trade-off relationship comprises assignment of a weight to each of the fields using the weight assigner 156. The weight may be used in weight summation over each of the entries.

The weight assigner 156 preferably comprises a user preference input which allows the user to define preferences between the fields. The weight assigner 156 may then select weights for assignment to the fields in such a way as to enforce the user defined preference.

The weight selection is preferably such as to maximize the user-defined preferences.

The fields themselves may be ordered preferentially, in which case the weight assigner may assign weights according to the position of the field.

The weight assigner includes a user input for receiving a parameter defining a number of entries of high desirability from the start of an ordered list, and the weight assigner may select weights to reinforce the desirability, in the same way as was discussed above regarding arbitrary user preferences.

The platform may additionally include a ranking expression unit 158 which is able to provide an expression basis for defining different types of ranking condition, for example a condition, a deviation, a deviation condition, a constraint, a simple trade-off relationship, a complex trade-off relationship, a preferred value within a range, a preferred range, a weighting. The expressions are combined by the user to form an objective function for use in ranking.

In a further embodiment of the present invention, there is provided a deal partitioner or deal allocator, whose purpose is to partition a desired purchasing deal by a buyer into sub-deals with possibly multiple sellers, that together strive to fulfill the buyer's needs. In some sense this is the opposite of the embodiment of Fig. 13. It is the side seeking the resources who has a catalog of resources. Either several resources are available from several sources or different resources are available from different sources.

The deal partitioner is preferably able to treat several cases of buyer's desired deals:

- a deal in which a single item type is desired (in a certain quantity).

- a deal in which several item types are desired (each in a certain quantity)

- a deal in which certain item combinations need be supplied by the same supplier.

- a deal in which the buyer's preferences, aside from quantities, are expressed using a goal program.

Sellers preferably have price tables for item(s) indicating prices per quantity ranges. Additionally or alternatively, sellers may have multi-item price tables, such that a seller is composed of sub-sellers. A sub-seller offers a collection of items, each with its own quantity range and price per unit, that must be bought together. In the following a distinction is made between sellers that can provide any number of items (unbounded sellers) and those that can only supply limited amounts (bounded sellers).

As will be discussed below in the Examples section, optimal partitioning algorithms, and heuristic approximation algorithms are provided, as well as combinatorial optimization approximation schemes in which a bounded deviation from an optimum solution is guaranteed.

The heuristic multi-item deal partitioning algorithm (MRG) is generalized to the case where instead of money (e.g., Dollars) cost is measured in objective function values of a goal program (the buyer's) and a package "price" is calculated based on the buyer's goal program and the quantities involved in each package as well as the quantities desired. The complete algorithm involves representing a buyer as a collection of sub-buyers and representing a seller as a collection of sub-sellers, each sub-seller, in turn, being associated with a seller goal program and a collection of explicit packages with specific quantities (rather than ranges). The complete algorithm also takes into account the total number of available (for sale) items of each kind that each seller has.

Examples

The examples section explains in greater detail, implementations of the above embodiments. These examples cover:

1. Automated and semi-automated negotiation platforms
2. Auctions and reverse auctions
3. Ranking and selection of alternatives

4. Construction and adaptation of business intentions, and
5. Deal splitting techniques.

The basic building block is a "Deal" which is composed of "Deal components", constraints, preferences and trade-offs. The deal components are given as a hierarchy of items (and sub-items) along with their attributes. The constraints represent real-world limitations and restrictions that must be adhered to while preferences indicate attribute values (or directions) that are more desired than others. Trade-offs are used to express situations in which decision makers are willing to change certain attribute values (thus giving up some benefits that are associated with these values) if other values are changed to compensate them for the lost benefit.

Deals are defined by users of the platform either through some user interface or through an Application Program Interface (API). The system then compiles the Deals into "Business Intentions". Each business intention contains a structure in which the various elements of the deal are expressed through mathematical terms and a "Goal Program" – an instance of the mathematical programming methodology that lies at the core of our system. The structure of the business intentions can be given in terms of trees where each node corresponds to an attribute of the deal. Goal programs are explained next.

The examples are organized as follows. The next section explains the basic terminology of E-contracts and the section that follows describes the concepts of goal programming. Section 1 describes the compilation techniques in the system; Section 2 presents the basic utilities (mostly variants of goal programs) that are used for mathematical manipulation of the business intentions; Section 3 discusses the special case of purchasing items from catalogs; Section 4 reviews the mechanisms that are used to express the negotiation tactics and strategies of the users and how they operate; Section 5 addresses the techniques we use to search, retrieve and rank information components that are stored in databases and Section 6 provides the details of deal-splitting techniques.

E-contract Terminology

We briefly review some of the terminology of E-contracts used in a previous patent application IL00/00516, the contents of which are hereby incorporated by reference. *Business Intentions* are made of *components*. Components may be atomic or compound (to any required depth). Components may also be inter-related (e.g., by containment, by edge or labeled-edge connection, or by arbitrary predicates). An important facet of a variable component is its possible association with one or more computational devices, although one-to-one association of a variable component with a computational device is particularly preferred, and is described herein. Such a computational device, based on its perceived state and messages, transforms a variable component into a component. The term "perceived state" is intended to include inputs, values of various components, values of certain other entities such as files, databases and the like. The "new" component is usually "more specific" than the variable component it replaces. Variable components and their associated computational devices embody transient or policy dependent aspects of the willingness to engage in a deal.

Forming an agreement, or negotiating a contract, requires the reconciliation of the constraints placed on deals by the (two or more) parties involved. Reconciliation involves forming an agreement or a contract, which, as much as possible, is subject to the directives of the parties, as well as to any general laws, which may apply. When examining two intentions, the process of reconciling the constraints may be considered to be a form of "fitting" to these constraints. Abstractly, this process fits the component structure of one party with the corresponding components of the other party.

In E-contracts, a component is represented as a *rooted labeled tree*. In fact, an intention is also a rooted labeled tree, which is composed of components, together with various constraints and computational devices. The most basic components are *simple atomic entities*, e.g., of type integer, float, string, etc. Next are *basic components* that are essentially (usually small) trees whose structure is agreed upon to represent a concept (e.g. car, sale, address). These basic components are called *classes* and they form the "words" of the common language. The word "class" hints at the fact that in an object oriented realization, these components are likely to be represented as object oriented classes, although the present invention is not limited to such a representation. A component may be a *variable component*. In this case it appears as a single node labeled with a typed variable. Such a type may be atomic, atomic list, class or list of classes. Such a variable component cannot exist in isolation but must be a leaf of a class.

Using classes, the parties compose their intentions, essentially forming "sentences" which in turn define possible deals. As noted, the purpose of an intention is to describe a deal that a party is willing to engage in. In E-contracts, the mechanism that composes words into sentences, or classes into intentions, relies on "variable instantiation" and the introduction of "operator nodes". A (leaf) variable component of an intention is optionally and preferably associated with a computation device, called a "commerce automaton" (CA) in this realization, which prescribes how the variable may be instantiated further during a later phase. A commerce automaton may outline a message exchange sequence between the parties. However, it should be noted that a commerce automaton is only one realization of a device or entity for exchanging messages between the parties and other realizations are possible as well. In addition to intentions, an e-commerce party also maintains *party information*, a database or file containing information relevant to the party's activities. This is part of the "system state".

A deal is manifested by creating a mutually agreed upon *electronic contract* (E-contract). The process of obtaining an E-contract begins with two initial intentions, presented by the parties. A formal process, called *unification*, a part of the realization of "fitting", is used to construct an agreed upon E-contract, provided such a contract is feasible. An e-commerce party may use unification to determine whether an E-contract is at all possible, prior to entering actual negotiations.

Goal Programming Models

Goal programming (GP) is an Operations Research methodology that was first proposed by Charnes and Copper in 1961 and has since proliferated into many research and application sub-areas. The GP methodology was inspired in part by the concept of "Satisficing" coined by Nobel Prize laureate Herb Simon – a term that refers to a combination of satisfying some objectives or constraints while sacrificing others. The basic idea of GP is that it is rather common to find constraints that are not stringent and many times decision makers are willing to accept an achievement level that is not exactly what they prefer most if they perceive that by this sacrifice they were able to satisfy other objectives or constraints. GP is an especially useful technique when users have multiple, and sometimes conflicting, objectives. GP provides two basic techniques for goal specification: prioritization and weighting (per priority level). Using these tools a user can indicate the relative importance of constraints, preferences and goals. In the GP literature prioritization is known as Lexicographic Goal Program (LGP) and weighting is referred to Weighted Goal Program (WGP). The main difference between these two techniques is in the formulation of the objective function. In WGP the objective is to minimize a weighted sum of deviations fro targets while in LGP the objective is structured as a hierarchy of levels where in each level there is a weighted sum of deviations. The program minimizes the first level; then it assigns the value obtained for the first level as a hard constraint and solves level 2; then it assigns the values obtained for both level 1 and 2 as hard constraints and solves level 3 and so on. LGP is our method of choice and we use it to express and manipulate deals in our contract management and negotiation system.

In our context, the general idea is to use the GP technique to represent the constraints and preferences of a deal. We shall first discuss how constituent elements are handled, then proceed to a simple intention, and finally expand on more complex intentions.

The general form of a GP program is as follows:

lexicographically_minimize {Expression 1, ..., Expression m} such that for $i=1,\dots,k$ we have *goal constraints*, g_i, of the form:

$c_i X + (D_i^-) \geq t_i$, or

$c_i X - (D_i^+) \leq t_i$, or

$c_i X + (D_i^-) - (D_i^+) = t_i$, and, in addition we have the constraint that

all D_i's ≥ 0, and, optionally,

some D_i's = o (indicating hard constraints)

The D_i variables are called *deviation variables;* those of the form (D_i^+) indicate an amount by which a goal is exceeded ("overshooting"), whereas those of the form (D_i^-) indicate under-achievement of goals. The Expression j's are called *minimization expressions.* The term lexicographically minimize (lex_min for short) implies an *order* on minimization, where the results of the D_i's, of minimizing up to *Expression i* are used as values in *Expression i+1, ..., Expression m.* So, the lower index expressions have a higher priority. Each Expression may refer to decision variables (X's), to deviation and other variables and to weights. Note that one may enforce hard constraints is by setting some deviation variables to zero. For example, to enforce a ≥ type constraint one may set $(D_i^-) = 0$.

Here is an example of a simple LGP, taken from reference [2].

Minimize: Priority 1$((D_1^-) + (D_2^-) + (D_3^+))$ *Priority 2*(D_4^+) *Priority 3*(D_5^-).

$$X_1 + X_2 + (D_1^-) - (D_1^+) = 30$$

$$X_3 + X_4 + (D_2^-) - (D_2^+) = 30$$

$$3X_1 + 2X_3 + (D_3^-) - (D_3^+) = 120$$

$$3X_2 + 2X_4 + (D_4^-) - (D_4^+) = 20$$

$$10X_1 + 9X_2 + 8X_3 + 7X_4 + (D_5^-) - (D_5^+) = 800$$

$$X_h, (D_i^-), (D_i^+) \geq 0 \quad i=1,\ldots,5,\ h=1,\ldots,4$$

We now explain how to transform the user's specifications into a GP. Then, we shall explain how to use such programs during negotiations.

1. Variables:

Variables are associated with atomic valued intention nodes. Variables are typed. The type may be integer or float. Variables may also be associated with discrete values as follows. Consider a variable that is associated with a color, which may be red, green, blue or yellow. Each color is associated with a value, e.g., red=1, green=2, blue=3 and yellow=4.

An important idea is that each variable is associated with a *default interval.* This interval is used for choosing default values. It also makes all variables *range-bounded.* The default interval may be a single point or a collection of ranges of values. Default values are also used when the constraints are not satisfied with the current set of variable assignments (this is an alternative to backtracking). Default intervals may be user specified or be derived from a database. A default interval is considered a hard constraint and is added to the other constraints. Choosing a value from a default interval may be done in a number of ways: minimize, maximize, percentage off maximum, average, random, etc.

2. Hard Constraints:

A hard constraint involving '<' or '>' is transformed into a hard constraint involving '≤' or '≥', respectively, by subtracting (respectively, adding) a small quantity. A hard constraint, of the form *expression* θ *value*, is compiled into $expression + (D-) - (D+) = value$ Depending on hardness, we may add constraints $(D+)=0$ for $\theta = '\leq'$ or $(D-)=0$ for $\theta = '\geq'$ and $(D-)=(D+)=0$ for $\theta = '='$. If hardness is more "limited" we may add a goal to minimize, of the highest priority, whose content is $(D-)+(D+)$. The understanding is that at the highest priority minimization expression should evaluate to zero. Alternatively, we may simply derive a goal of the form $LARGE^*(D-)$, or $LARGE^*(D+)$ or $LARGE^*(D-) + LARGE^*(D+)$ and treat it according to its weight. This latter form increases feasibility of a solution. Here *LARGE* is a sufficiently large value in the domain considered.

3. Soft constraints:

A soft constraint of the form *expression* θ *value* is compiled into

$expression + (D-) - (D+) = value.$

For example, consider the constraint *soft (Qty=20)*. It is compiled into

$Qty + (D1-) - (D1+) = 20,$

Here, again, we use *deviation variables*. In fact, such constraints express preferences, namely being close to the target. In case a deviation in the direction (D1-) is, say, four times as undesirable as a deviation in the direction *(D1+)*, then:

Case 1: The soft constraint is assigned a priority and it is the only one at its priority level. This means we should minimize $4(D1-) + (D1+)$.

Case 2: Otherwise, we need to normalize this constraint so that we can compare it to other constraints. We use the idea of *percentage deviation* where the percentages are based either on the target value (as demonstrated below) or on the range of permissible values for the relevant attribute (as we show later in Chapter 1).

1. Define $(Dp1+) = (D1+)/target$ and $(Dp1-) = (D1-)/target$.

2. What we minimize is the expression *minimize (A*(Dp1-) + (1-A)*(Dp1+)*, *[0≤A≤1]* e.g., *minimize (0.80*(Dp1-) + 0.20*(Dp1+))*.

3. If, in addition, the overall weight of this soft constraint is *W*, then this value is associated with the most undesirable deviation and the weight for all other deviations are smaller than or equal to W. For example, we may solve the expression *minimize W* (Dp1-) +0.20 * W* (Dp1+)* when negative deviations are considered 5 times as important as positive deviations.

Now consider a constraint of the form *expression ≤ value.* It is compiled as above into:

expression +(D-) – (D+) = value.

Here the goal to minimize is *W*(Dp1+)*, where *W* and *(Dp1+)* are as above. The cases of *θ*='≥', '>', '<' are handled similarly.

4. Preferences:

We allow minimization (*min*) or maximization (*max*) preferences on functions, e.g., *min 2*X+5*Y*. We can also give such preference a *weight*, indicating its importance. For example (*W1 is the weight)*: *W1: minimize: 2*X+5*Y*.

This preference is compiled as follows. First, an "optimistic" yet "reasonable" target for the minimization is determined (by using default intervals, user specification or solving a simplified linear program). For example, if a reasonably optimistic small value for the above expression is *100*, the preference is restated as the soft constraint: *W1: 2*X+5*Y≤100*.

From this point on, it is treated as an ordinary soft constraint. As stated, the value, *100* in the example may be determined by using another linear program with the minimization objective as the only objective.

Preferences can also be applied to variables corresponding to discrete values. For example, suppose we prefer red, green, blue and yellow in that order. Further, suppose we rate our preferences on a scale from 1 to 100. We can model this using the *soft* constraints:

100: Color=1

50: Color=2

20: Color=3

20: Color=4

We also add the hard constraints:

$Color \geq 1$

$Color \leq 4$

integer(Color)

This formulation of soft constraints will favor the more preferred targets.

Recall that in general we have a number of preferences, each translated into a soft constraint, say $P_0, \ldots, P_k$. These, and the "original", soft constraints are partitioned into a number of priority levels. Priority levels are handled one by one using lexicographic minimization. Conceptually, the results of minimization at level *i, that is, minimization expressions of higher priority,* are inserted as constraints in the minimization at level *i+1*. Consider again the program in the example above. If we solve it using a linear programming package, we first present the highest-level linear program:

Minimize: (Expression of Priority 1) $((D1^-) + (D2^-) + (D3^+))$

$X1 + X2 + (D1^-) - (D1^+) = 30$

$X3 + X4 + (D2^-) - (D2^+) = 30$

$3X1 + 2X3 + (D3^-) - (D3^+) = 120$

$X1, X2, X3, (D1^-), (D1^+), (D2^-), (D2^+), (D3^-), (D3^+) \geq 0$

$(D1^-), (D1^+), (D2^-), (D2^+) \leq 30$

$(D3^-), (D3^+) \leq 120$

Solving, we discover that the result is $((D1^-) + (D2^-) + (D3^+)) = 0$. We therefore form the next level linear program as:

Minimize: (Expression of Priority 2) $1(D4+)$

$X1 + X2 + (D1^-) - (D1^+) = 30$

$X3 + X4 + (D2^{-}) - (D2^{+}) = 30$

$3X1 + 2X3 + (D3^{-}) - (D3^{+}) = 120$

$((D1^{-}) + (D2^{-}) + (D3^{+})) = 0$ *(This is the "newly fed" constraint.)*

$3X2 + 2X4 + (D4^{-}) - (D4^{+}) = 20$

$X1, X2, X3, X4, (D1^{-}), (D1^{+}), (D2^{-}), (D2^{+}), (D3^{-}), (D3^{+}), (D4^{-}), (D4^{+}) \geq 0$

$(D1^{-}), (D1^{+}), (D2^{-}), (D2^{+}) \leq 30$

$(D3^{-}), (D3^{+}) \leq 120$

$(D4^{-}), (D4^{+}) \leq 20$

The solution turns out to be $(D4^{+})=10$. This is "fed" into yet one more (f) linear program that optimizes $10X1+9X2+8X3+7X4$.

The remaining issue is how to compile a single priority level.

There are two basic methods for compiling objective levels, we generally use Method 1 and for the sake of completeness mention Method 2 as a possibility.

Method 1: All the soft constraints within a priority level are compiled into a single expression to be minimized, by summing up the individual minimization expressions compiled for each soft constraint in isolation.

Method 2: Use the *min-max* [8] idea of treating each soft constraint in isolation and then bounding the maximum deviation on any particular soft constraint. Assume we have a total of K minimization expressions corresponding to K soft constraints at priority level j. This is compiled into an overall minimization objective for level j:

min Q

expression part of minimization expression $1 \leq Q$

expression part of minimization expression $K \leq Q$

Observe that the result will tend to minimize more the important soft constraints, that is, those with larger weights. The advantage in *min-max* is more flexibility and minimization of missed goals. Observe that each minimization expression is already percentage-wise and weight-wise normalized. There are advantages and disadvantages to each method. We can preferably run the user's intention in parallel in two versions, one using Method 1 and one using Method 2. We can also decide on Method i (i=1 or i=2) as default and allow the user to change it for a particular run.

Introduction

This section details the compilation techniques that are used to translate the user's inputs into a goal-programming (GP) model. The compilation techniques are geared to cover all the compilation aspects in a way that isolates the user interface from other system layers

The overall conceptual architecture is as follows. Business intentions, i.e., deals, are specified in the graphical user interface (GUI) layer. We note that the term 'GUI layer' also encompasses accesses of a software agent or through an API by a program or a software agent. This specification includes the various products/services that are desired along with constraints, preferences, (i.e., targets) and trade-offs. In addition, the various preferences and trade-offs are weighted. That is, their relative importance is indicated. While GUI hints at a user interface, this layer may also correspond to such information as used by a computerized agent as opposed to a human, or even a human and agent combination.

The GUI level specifications are compiled into formal objects called business intentions. These intentions are used to match deals, and later on to negotiate deals in 1-1 or 1-n modes. One component of a business intention encodes constraints, preferences, and trade-offs. This component is in the form of a goal program (GP). Goal programs are a well-known formalism for representing multiple, and often conflicting, objectives. The subject of this document is to outline the methods, by which user intentions are translated into goal programs.

LP package requirements

In order to perform some of the techniques in the following sections, the LP package is required to have certain capabilities listed below. These features are usually found in commercial packages available today.

- Ability to handle integer programming, this is used to:
    a. Define binary flags.
    b. Define discrete-value variables.
- Epsilon techniques are used for OR compilation.

Input for compilation

The sections below describe how we compile the data provided from the GUI layer. However, the compilation, described below, is not affected by the source of the input data. Moreover, these sections describe what should be provided at some outer level in order to compile a problem properly, so that it can be used in the other system layer. For example, a certain compilation technique may expect to receive some weights as inputs. The user through the GUI could have entered these weights or, they might have been computed by some procedure at a higher layer. At any rate, the compilation technique is not affected by the way these weights are generated.

Ordinary goals

Basic Goal representation

This section deals with *single dimension point goals*, the most basic type of goal constraints. Reference is now made to Fig. 18 which is a simplified diagram of a single dimensional point goal.

Input for compilation

(*Regarding the decision variable* y)

- Target value: T_y.
- Level of objective function: L.

- Weights for lower and upper deviations from the target T_y: W_y^-, W_y^+. To make it easy to combine objectives later on, we require, without loss of generality, that the maximum of W_y^-, W_y^+ equals 1.0. No generality is lost since the relative importance of deviating in these directions can be easily encoded within this restriction.

- Relative importance of this variable y within the objective function level L : V_y.

Representation in the GP problem

Given the above inputs, we add to the GP problem a goal constraint to express the user's desire to reach the target value T_y.

Goal constraint: $y+\delta^- -\delta^+ = T_y$

(1)

We also express the importance of reaching the target goal by adding a term containing weighted deviations to the suitable level (L) of the objective function.

Objective (at level L): $\mathit{Min} \quad Z = V_y \cdot \{W_y^- \cdot \delta^- + W_y^+ \cdot \delta^+\} + \ldots$

(2)

Comments

1. Notice that the weights W_y^-, W_y^+ are penalty factors, representing how much the user dislikes a deviation in either direction.

2. Targets may appear at the extreme end of the interval defined by the lower and upper bounds for the variable y. In that case, only one deviation variable (pointing inwards into the interval) is meaningful. Therefore, only one deviation variable will participate in the objective function. However, such cases may indicate poor modeling. Therefore, it is recommended that the UI will apply ways to (gently) discourage the user from getting there.

Advanced Goal representation

Strong one-sided goals

In these cases the user provides a point-target value T (within the [L,U] bounds specified for the attribute) and specifies a linear penalty function (through a fixed weight per unit of deviation) only on one side of the target whereas he is truly indifferent with regard to the other side. For example, a buyer who wishes to receive a product no later than 10 days from the contract date and is totally indifferent to receiving it earlier. This means that we have a target range that starts at one of the bounds and ends at the given point-target as depicted in the following chart. Reference is herein made to Fig.19 which is a simplified diagram of a strong one-sided goal.

This case is modeled through the following formulation (which can easily be generalized to multi-attribute cases):

$$\begin{aligned} &Min \quad W^{+} \cdot \delta^{+} \\ &s.t. \\ &\quad X + \delta^{-} - \delta^{+} = T \\ &\quad L \leq X \leq U \\ &\quad other\ constraints \end{aligned}$$

Weak one-sided goals

The only difference between the "weak" and the "strong" cases is that in the former the user does have a preference for values on the side of the point-target for which no penalty was specified in the strong case. Using the same example, a 10-day delivery target is a plausibly outcome for the user but if possible, he would prefer to receive the product as early as possible. We further assume that the user is unable to quantitatively state the preference on the undefined side of the target. Hence, the "pure" (or "theoretic") target is the bound (L in our example), while T is a plausible target. Reference is made to Fig. 20 which is a simplified diagram of a weak one sided goal.

This case is handled through Hanan's procedure.

$$\begin{aligned} &LexMin && \{(W^{+} \cdot \delta^{+}),(\delta^{-})\} \\ &s.t. && \\ & && X+\delta^{-}-\delta^{+}=T \\ & && L \leq X \leq U \\ & && \textit{other constraints} \end{aligned}$$

Complex single-sided goals

Unlike the previous case, here the user is capable of specifying the degree by which he prefers deviations from the theoretical target up to the plausible one vis-à-vis the deviations from the plausible target. This case is depicted in appended Fig. 21.

Notice that we could have modeled the penalty function in the interval [L,T] as a "bonus" by raising the horizontal axis upwards until it intersects with the penalty function at T (changing the definition of zero penalty). This would not change anything in the outcomes of our formulation.

$$\begin{aligned} &Min && W_1^{+} \cdot \delta_1^{+} + W_2^{+} \cdot \delta_2^{+} \\ &s.t. && \\ & && X+\delta^{-}-\delta_1^{+}-\delta_2^{+}=L \\ & && \delta_1^{+} \leq T-L \\ & && L \leq X \leq U \\ & && \textit{other constraints} \end{aligned}$$

Simple two-sided goals

This is the classical case in which the user specifies a point target with penalties on deviations in both directions (not necessarily with the same weights). Mathematically, we formulate it as:

$$\begin{aligned} &Min \quad W^{-}\cdot\delta^{-}+W^{+}\cdot\delta^{+} \\ &s.t. \\ &\qquad X+\delta^{-}-\delta^{+}=T \\ &\qquad L\le X\le U \\ &\qquad \textit{other constraints} \end{aligned}$$

Two-sided goals with interval range

Similar to the previous case where instead of a point-target we have a target range as depicted in the appended Fig. 22.

This case is formulated by:

$$\begin{aligned} &Min \quad W^{-}\cdot\delta_1^{-}+W^{+}\cdot\delta^{+} \\ &s.t. \\ &\qquad X+\delta_1^{-}+\delta_2^{-}-\delta^{+}=T_2 \\ &\qquad \delta_2^{-}\le T_2-T_1 \\ &\qquad L\le X\le U \\ &\qquad \textit{other constraints} \end{aligned}$$

Complex two-sided goals (segmented U-shaped functions)

This is the most general case that we can treat at the time this document is being written. It encompasses all the cases presented earlier as special cases. It contains target range as well as multi-slop penalty functions on both sides of the range. A simple illustration of a case with two slopes on each side is depicted in appended Fig. 23.

This case is solved through

$$\begin{aligned} &Min \quad W_1^{-}\cdot\delta_1^{-}+W_2^{-}\cdot\delta_2^{-}+W_1^{+}\cdot\delta_1^{+}+W_2^{+}\cdot\delta_2^{+} \\ &s.t. \\ &\qquad X+\delta_1^{-}+\delta_2^{-}+\delta_3^{-}-\delta_1^{+}-\delta_2^{+}=T_2 \\ &\qquad \delta_2^{-}\le T_1-T_0 \quad , \ \delta_3^{-}\le T_2-T_1 \ , \ \delta_1^{+}\le T_3-T_2 \\ &\qquad L\le X\le U \end{aligned}$$

other constraints

Weights

In the previous section we introduced a goal into the objective function using three values V_y, W_y^-, W_y^+.

The value V_y expresses the relative importance between the deviation variables of the goal, and the other goals introduced at the same level.

The values W_y^- and W_y^+ express the relative importance of deviating from the target value t_y. To maintain consistency across the goals in the same priority level we assign the value of 1.0 to the weight associated with the more severe deviations and a value that is smaller than or equal to 1.0 to the other weight. Thus, if a deviation in the worst direction occurs, the entire V_y will be applied. But, if the deviation occurs in the other direction, only a certain portion of V_y will be in effect.

Bounds

Here we describe hard bounds on the problem *variables*, both decision and deviation ones. We do not address the issue of hard bounds in general. We require that all the GP problems that we compile be bounded so as to avoid unbounded or unrealistic solutions. Bounding the problem has other important advantages. These include, easier design of the utility layer, (such as, calculating the *worst* solution), and easier compilations (e.g., performing the scaling described above), and more efficient run-time of the LP package.

Decision variables bounds

As mentioned above, decision variables get their bounds either through the UI level, or by interacting with the user and the Database or Catalog, and are out of the scope of this section.

Deviation variables bounds

Suppose we have a goal constraint of the form $\{g_i + \delta_i^- - \delta_i^+ = T_i\}$.

In general, we set the bounds on the deviation variables δ_i^- and δ_i^+ here as follows:

$$\begin{aligned} lb(\delta_i^-) &= 0 & \quad and \quad & lb(\delta_i^+) = 0 \\ ub(\delta_i^-) &= T_i - lb(g_i) & & ub(\delta_i^+) = ub(g_i) - T_i \end{aligned} \tag{3}$$

Notice that ub(f) is the upper bound of f, and lb(f) is the lower bound of f.

When we calculate the bounds of a goal of the form $g_i = \sum a_r X_r$ as follows:

$$\begin{aligned} ub(g_i) &= \sum (a_r \cdot ub(X_r)) \\ lb(g_i) &= Min\{a_r \cdot lb(X_r)\} \end{aligned} \tag{4}$$

Reference in the above regard is made to Fig. 24.

Scaling

The scaling of a GP constraint aims to handle two problems. (See Romero, 1991, pp. 35-36). The first problem is concerned with goals that are expressed with very different numerical values (e.g. price in millions of dollars vs. number of days for delivery). In such cases we face difficulties in combining the corresponding deviation variables within the same priority level in the objective function.

The second problem is concerned with objective functions that mix deviations that are associated with both discrete and continuous variables. To express ordinal preferences in the dimension of a discrete variable we use internal weights that are arbitrary with respect to the other variables that share the same level of the objective function.

Scaling by Targets

As mentioned above, scaling is concerned with representing all the values in the objective function in consistent units. Given a goal constraint $\{g_i + \delta_i^- - \delta_i^+ = T_i\}$ we scale the deviation variables such that they express the amount of *relative* deviation from the target values of each goal, as follows:

$$\{g_i + \delta_i^- - \delta_i^+ = T_i\} \quad \text{, when } 0 \le |T_i| \le 1 \quad \text{(i.e., goal is not scaled)}$$

$$\left\{\frac{g_i}{T_i} + \delta_i^- - \delta_i^+ = 1\right\} \quad \text{, when } |T_i| > 1 \qquad (5)$$

For example, suppose we have two goals

$$\{Y + \delta_y^- - \delta_y^+ = 5000\}$$

$$\{X + \delta_x^- - \delta_x^+ = 100\}$$

The scaling described above, will assign the same numerical value to a deviation of 500 in Y and a deviation of 10 in X, (i.e., a 10% deviation in both cases).

Handling bounds

In section 0 we discuss the upper and lower bounds that are defined for all the variables in a GP problem. However, after the scaling presented above, we should update the bounds for the deviation variables. Scaling the bounds, similarly to the scaling above, as follows, does this:

$$lb(\delta_i^-) = 0 \qquad \textit{and} \qquad lb(\delta_i^+) = 0$$

$$ub(\delta_i^-) = \frac{T_i - lb(g_i)}{T_i} \qquad ub(\delta_i^+) = \frac{ub(g_i) - T_i}{T_i} \qquad (4')$$

Comments

1. Currently we ignore constraints with negative targets at the right hand side (RHS) of the goal. However, for the sake of completeness the scaling is represented in the most general way

2. The deviation variables are apparently not affected by the scaling. However, they can be considered as new deviation variables with the same name but different semantics

3. Notice that the scaling of bounds described above is only affected for deviation variables (i.e., the bounds on decision variables are not changed).

Example

Suppose we have the following goal constraints (i.e., we'd like X_1 to be close to 100 but we do not care if it exceeds 100, X_2 close to 1 but we don't care if it is larger than 1, X_3 close to 0.9 but we do not care if it's below 0.9, and X_4 close to 40 and we don't care if it's below 40). Further, suppose that we consider $X_1,\ldots,X_4$ equally important. Now, writing the objective to minimize $\left(\delta_1^- + \delta_2^- + \delta_3^+ + \delta_4^+\right)$ would be mixing apples with oranges, due to the ranges of these variables. So, we scale the goal constraints as showed in the right column below. Once we do that, we can write the goal, as $\left(\delta_1^- + \delta_2^- + \delta_3^+ + \delta_4^+\right)$. Note that what we have done is to interpret the fact that the variables are equally important as meaning that percentage-wise deviations in achieving these goals are equally important. Had we been told that X_1 is twice as important as the other variables, the resulting goal would have been $\left(2 \cdot \delta_1^- + \delta_2^- + \delta_3^+ + \delta_4^+\right)$. Finally, we have not altered the goal constraint with 0.9 as it is reasonably close to 1.

GP problem (before scaling)

$$\begin{aligned} X_1 + \delta_1^- - \delta_1^+ &= 100 \\ X_2 + \delta_2 - \delta_2^+ &= 1 \\ X_3 + \delta_3^- - \delta_3^+ &= 0.9 \\ X_4 + \delta_4^- - \delta_4^+ &= 40 \end{aligned}$$

Bounds:

$$\begin{aligned} 0 &\le X_1 \le 200 \\ 0.3 &\le X_2 \le 2 \\ 0 &\le X_3 \le 2 \\ 25 &\le X_4 \le 50 \end{aligned}$$

$0 \le \delta_1^- \le 100,\ 0 \le \delta_1^+ \le 100$

The same GP problem (after scaling)

$$\begin{aligned} 0.01 \cdot X_1 + \delta_1^- - \delta_1^+ &= 1 \\ X_2 + \delta_2^- - \delta_2^+ &= 1 \\ X_3 + \delta_3^- - \delta_3^+ &= 0.9 \\ 0.025 \cdot X_4 + \delta_4^- - \delta_4^+ &= 1 \end{aligned}$$

Bounds:

$$\begin{aligned} 0 &\le X_1 \le 200 \\ 0.3 &\le X_2 \le 2 \\ 0 &\le X_3 \le 2 \\ 25 &\le X_4 \le 50 \end{aligned}$$

Scaling by intervals

Scaling can be done in interval terminology rather than by target. The advantage is that we can naturally handle a target of 0. For example suppose that *D* measures delay and we prefer zero delay. Suppose that D is in the range [700,850]. Suppose the target is 800. Observe that the question to which the answer is 5 is "what penalty would you assign to a deviation the size of the whole interval?" Of course, at the GUI level, the question may be phrased differently.

$$
\begin{array}{lll}
\mathit{Min} & 5.\delta_p^- + \ldots & /^{*}\ 5\ \textit{is penalty for a deviation of}\ 150\ \textit{units}\ ^{*}/ \\
\mathit{s.t.} & & \\
 & \frac{D}{150} + \delta_p^- - \delta_p^+ = \frac{800}{150} & /^{*}\ \textit{deviation measured as fraction of interval}\ ^{*}/ \\
 & 700 \leq D \leq 850 &
\end{array}
$$

Pre- and post-matching specification via intervals

A user specifies the importance in "interval units" prior to matching with other parties. The effective interval is the intersection of the intervals specified by the parties. Observe that when specifying, a party does not know which other parties' intervals will be encountered. We differentiate between resulting point intervals, small intervals, and normal intervals.

Normal intervals

Suppose a party specified an importance of 5 and its interval at specification time had 150 units as in the example above. Suppose the size of the intersection with another party's interval has just 50 units. First, if the "old" target is outside the intersection (e.g., the intersection is [700,755]), we "push it" towards the closest intersection boundary (755) thereby creating a new target. If the "old target" is within the new interval (e.g., the intersection is [760,810]), we leave the target unchanged. Next, we modify the goal constraints and objective function to reflect the new interval. The resulting compilation, assuming a new interval of [700,755] and following the scheme above, is depicted below. Observe the changes in the target in the goal constraint and the new bounds on the interval. Also note the addition of the term 5(45/150) to the objective function due to a target shift from 800 to 755. This constant term does not affect optimization and we may remove it in treating the goal program. It is brought back only when the original goal program is used for ranking offers. Alternatively, we can keep the constant term, as it's important during negotiations that a big "price" is paid.

$$\begin{array}{lll} Min & 5.\frac{45}{150}+5.\delta_p^-.+\ldots & /\text{* 5 is penalty for a deviation of 150 units */} \\ s.t. & & \\ & \frac{D-705}{150}+\delta_p^- - \delta_p^+ = \frac{755-705}{150} & /\text{* deviation measured as fraction of interval */} \\ & 705 \le D \le 755 & \end{array}$$

Note that we can do away with the −705 term that appears in both sides.

Point intervals

Point intervals are simply points, namely the intersection of the parties' interval is a point. In this case, we substitute for the variable in question its value (the point) in the constraints. This also gives values to the deviations. We introduce the resulting constants into the objective functions. As before, we have the option of removing these constants altogether.

Small intervals

The resulting interval may be "small". However, smallness is a relative term and what is small for party A may be large for party B. If both parties agree that the interval is small, then they may simply choose a mid-point between their targets (shifted into an interval boundary point if needed) and we are back to the case of a point interval. If the interval is large for both we treat it as normal. If one party considers it small and the other as big, we treat it as normal.

Transforming from target-based to interval-based formulation

In the target-based compilation method we treat importance factor as related to fractional deviation from a target of one. Consider an attribute P (for price) with a domain of [1,1000] and target of 500. Further suppose it has importance level of 7 and that only positive deviations matter. Scaling by targets would lead to:

$$\begin{array}{ll} Min & 7 \cdot (\delta_P^+) + \ldots \\ s.t. & \\ & \frac{P}{500} + \delta_P - \delta_P^+ = 1 \\ & 1 \le P \le 1000 \end{array}$$

Suppose that following unification with another party's business intention, the revised domain is [450,750]. Assume further that the other party's target for P is 700. Let us assume that we want to treat this new domain in such a way as to give an equal importance to a Dollar in the vicinity of 500 as to a Dollar in the neighborhood of 750. To do that, we measure deviations in 'interval units' (before it was in 'target units'). So, the compilation is transformed into:

$$\begin{array}{ll} Min & \frac{7.300}{500} . \delta_P^+ . + \ldots \\ s.t. & \\ & \frac{P}{300} + \delta_P^- - \delta_P^+ = \frac{500}{300} \quad \text{/* deviation measured as fraction of int erval */} \\ & 450 \le P \le 750 \end{array}$$

The factor ((7 * 300)/500) is due to the fact that an importance of 7 was attributed to a deviation of 500 Dollars (the target size), which translates to an importance level of (7/500) for a one Dollar deviation. In the "new" version, deviation is measured as a fraction of the interval size, so to translate it to Dollars we multiply by 300. Hence the resulting factor in the objective function. So, what we showed is how to move from 'target oriented' compilation to 'interval oriented compilation'.

What we showed so far is that we can represent the user 's preference in terms of intervals and that we can translate from target based representation to interval-based representation. The other direction is not always possible due to targets of zero.

Targets outside of their domain

Occasionally, we may encounter target values that are outside of the feasible range of values specified by the GP for a given variable. This may happen, for example, when one of the negotiating parties defined in his intention a range that is much larger than the range defined by the other party and a target that is closer to the high end of his range (say, a range of [100,600] with a target at 500 for a Price (P) variable while the other party's range is [100,120] with a target at 110). The unification procedure sets the feasible range for this variable according to the intersection of the ranges – in our case this means that it is set according to the definition of the second party.

These situations may cause severe distortions in the evaluation of the deviation from the goals. When weights were solicited in order to be used as coefficients in the objective function, the users were thinking about the relative negative effect of deviation by one unit above or below a target in one dimension versus another. In the case described above, the proportional deviation in P for the first party will range between 0.76 to 0.8 $\left(\frac{380}{500},\frac{400}{500}\right)$. Hence, the importance of this deviation is rather marginal since the deviation itself is quite significant no matter which value will eventually be selected for P.

To correct these deficiencies we set the bound nearest to the target as an *effective target* for a variable whose original target is outside its allowed domain (in our case, 120 becomes the effective target). Consequently, we need to transform the original deviation variables into new ones. This is demonstrated through the following example.

Original formulation

$$\begin{aligned}
&Min \quad 7\cdot\left(\delta_P^-\right)+5\cdot\left(\frac{\delta_D^-+\delta_D^+}{2}\right)\\
&s.t.\\
&\qquad \frac{P}{500}+\delta_P^- -\delta_P^+ =1\\
&\qquad \frac{D}{80}+\delta_D^- -\delta_D^+ =1\\
&\qquad 100\le P\le 120\\
&\qquad 30\le D\le 90
\end{aligned}$$

We define a transformation between the original deviation (δ_P^-) and a new one ($\hat{\delta}_P^-$) that would relate the deviation to the effective target: $\frac{120\cdot\hat{\delta}_P^- +380}{500}=\delta_P^-$. This leads to the following revised model.

Transformed formulation

$$\begin{aligned}
&Min \quad \frac{7\cdot 120}{500}\cdot\left(\hat{\delta}_P^-\right)+5\cdot\left(\frac{\delta_D^-+\delta_D^+}{2}\right)\\
&s.t.\\
&\qquad \frac{P}{120}+\hat{\delta}_P^- -\hat{\delta}_P^+ =1\\
&\qquad \frac{D}{80}+\delta_D^- -\delta_D^+ =1\\
&\qquad 100\le P\le 120\\
&\qquad 30\le D\le 90
\end{aligned}$$

Notice that the revised weight on deviations from the effective target in P is now much smaller than its original value. Also notice that the fixed term $\left(\frac{7\cdot 380}{500}\right)$ was omitted from the objective function, as we do not carry constants in these functions. (Note that we still have to "remember" this constant once we compare the end result of negotiations with this GP with an end result of negotiating with another GP obtained similarly from the original GP but for another interval.) The treatment of the other variable (D) was not directly affected by the transformation. Indirectly, however, the weight on the deviations from its target, which was inferior before, is now superior to that associated with deviations in P. This is the right thing to do as indeed all points in [100,120] are roughly of the same quality regarding a target of 500.

The general form for deriving the new deviation variable $\hat{\delta}_p^-$ is:

$$\frac{ET \cdot \hat{\delta}_p^- + |OT - ET|}{OT} = \delta_p^-$$

where OT is the "old" target and ET is the new effective target (an end-point of the unified interval). When the original target is greater than the upper bound, ET is equal to the upper bound of the variable and $|OT - ET| = OT - ET$. In the other direction, ET is equal to the lower bound and $|OT - ET| = ET - OT$.

Of course, the goal constraint that was originally compiled using OT is now replaced with one using ET and $\hat{\delta}_p^-$.

Cases where OT is smaller than 1 and ET is larger than 1

When OT is smaller than 1 the original goal constraint was not normalized. Hence, in these cases the proper transformation is:

$$ET \cdot \hat{\delta}_p^+ + (ET - OT) = \delta_p^+$$

Notice that δ_p^+ is expressed in absolute terms while $\hat{\delta}_p^+$ is expressed in relative terms.

Cases where OT is greater than 1 and ET is smaller than 1

Here we go in the opposite direction. The original goal constraint was normalized and the transformed one is not. Hence, here δ_p^+ is expressed in relative terms while $\hat{\delta}_p^+$ is expressed in absolute terms and,

$$\frac{\hat{\delta}_p^+ - (OT - ET)}{OT} = \delta_p^+$$

Trade-off relations

This section deals with advanced goal constraints, which aim to better express the user preferences.

Single dimension interval goal

A single dimensional interval goal is shown in Fig. 25.

Input for compilation

(Regarding the decision variable y)

Target interval: (L_y, U_y)

Level of objective function: L

Relative importance within level: V_y

Weights for lower and upper deviations within the y dimension: W_y^-, W_y^+, here too, without loss of generality, we assume that the max of W_y^-, W_y^+ is 1.0.

Representation in the GP model

Goal constraints:

$$y+\mu_1-\delta^+=U_y$$
$$y+\delta^- -\mu_2=L_y$$

Objective (at level L):

$$\text{Min} \quad Z=V_y\cdot\{W_y^-\cdot\delta^- +W_y^+\cdot\delta^+\}+\ldots \tag{6}$$

Comments

1. μ_1, μ_2 are auxiliary variables that do not enter the objective function. As in the other GP constraints, there is no need to impose the non-linear constraints $\mu_1\cdot\delta^+=\mu_2\cdot\delta^-=0$ since they are satisfied automatically. This phenomenon is due to the fact that the two columns corresponding to δ^+ and δ^- in the matrix of coefficients for the linear program are linearly dependent. Since the optimal basis of linear programs does not consist of linearly dependent columns we know that both variables cannot participate in

such basis (The variables corresponding to the optimal basis are the only ones to *receive non-zero* values in an optimal solution).

2. Calculating bounds for the problem's variables is very similar to the case of ordinary goals (see 0).

I.e., $0 \le \delta^- \le L_y$, $0 \le \delta^+ \le U_y$, *and* $0 \le \mu_1, \mu_1 \le U_y - L_y$

Two dimensional trade-off goal

A two dimensional trade-off goal is illustrated with reference to Fig. 26.

Input for compilation

Two equally preferred points: $(x_1, y_1), (x_2, y_2)$, which determine – the relations (all other variables held fixed):

$$b = \frac{y_2 - y_1}{x_2 - x_1} \quad ; \qquad a = y_1 - \frac{y_2 - y_1}{x_2 - x_1} \cdot x_1$$

Level of objective function: L

Importance within level: V

Relative weights for lower and upper deviations from the trade-off line: W^-, W^+. Here too, without loss of generality, we assume that the max of W_y^-, W_y^+ is 1.0.

Representation in the GP model

Goal constraints: $y - b \cdot x + \delta^- - \delta^+ = a$

Objective (at level L): $Min\,Z = V \cdot \{W^- \cdot \delta^- + W^+ \cdot \delta^+\} + \ldots$ (7)

Comments

1. Trade-offs are preferably only amongst variables associated with the same priority level in the objective function. Otherwise, logical inconsistencies concerning levels' importance may result.

2. These trade-offs raise the question of allocating a "finite capital" of weight values (see 0). That is, suppose the user intended to

associate identical importance with three decision variables; x, y, z. Let's further say that they are all measured by the same units (say, in dollars). Nevertheless, since Z will be represented in the objective function only with its individual weights while x and y will have in addition to their individual weights the weights associated with their trade-off, their relative weight in the objective is going to be higher.

3. In compiling trade-offs we can scale them either "by target" or "by interval" (see the Scaling section in the basic goal compilation). Scaling of trade-offs by target is done by dividing the trade-off constraint by the value of the intercept a (leading to 1 on the right-hand-side). To scale by interval we divide the trade-off constraint by the interval associated with the dependent variable (y in the examples above). This is done because the deviations in the trade-off constraint are expressed in its units.

4. We do not put tight bounds for the deviation variables; instead, we put some reasonable bounds according to the bounds of the decision variables:

$$0 \leq \delta^-,\ \delta^+ \leq Max\{y(lb(x)), y(ub(x))\}$$

where we define $y(\mathrm{ub}(x)) = a + b \cdot ub(x)$, and $y(\mathrm{lb}(x)) = a + b \cdot lb(x)$. This expression insures a good upper bound without depending on whether the slope, determined by the sign of *b*, is increasing or decreasing.

Single dimension two-point goal

A single dimensional two-point goal is illustrated with reference to Fig. 27.

Input for compilation

Two equally preferred targets: T_1, T_2

Level of the objective function for both targets: L

Relative importance within level: V

Relative weights for lower and upper deviations within the dimension: W^-, W^+. Here too, without loss of generality, we assume that the max of W_y^-, W_y^+ is 1.0.

Representation in the GP model

Goal constraints:

$$\begin{aligned} y+\delta_1^- - \delta_1^+ &= T_1 \\ y+\delta_2^- - \delta_2^+ &= T_2 \end{aligned} \tag{8}$$

Objective function (at level L):

$$Min \quad Z=\left\{W^- \cdot Min\left\{\delta_1^-, \delta_2^-\right\} + W^+ \cdot Min\left\{\delta_1^+, \delta_2^+\right\} + Min\left\{W^+ \cdot \delta_1^+, W^- \cdot \delta_2^-\right\}\right\}$$

Comments

1. In the objective function above, the first term corresponds to the region to the left of point T_1, the second to the region to the right of T_2, and the third to the middle region.

2. The objective function above does not lend itself easily into a linear programming formulation. Hence, at the current version of the software we shall break these situations into two intentions. The control mechanism that builds and releases intentions will be required to recall the XOR relationship between them.

3. The weights associated with negative and positive deviations from the two targets do not necessarily have to be identical (as presented in the figure above). In such cases, we'll have to rewrite the equation, e.g. instead of writing W^- we write $W_{index(Min\{\delta_1^-, \delta_2^-\})}$.

4. Calculating the bounds of the deviation variables is as describes in *0* for Ordinary goals. Notice that this trade-off consists of two ordinary goals.

. Piecewise linear two dimensional trade-off goal

A piecewise linear two-dimensional trade-off goal is illustrated with respect to Fig. 28.

Input for compilation

Three equally preferred points: $(x_1, y_1), (x_2, y_2), (x_3, y_3)$ (all other variables held fixed)

Level of the objective function for both targets: L

Weights: W^-, W^+, here too, without loss of generality, we assume that the max of W_y^-, W_y^+ is 1.0.

Representation in the GP model

Goal constraints:

$$y - b_1 \cdot x + \delta_1^+ - \delta_1^- = a_1$$

or

$$y - b_2 \cdot x + \delta_2^+ - \delta_2^- = a_2$$

Objective function:

$$Min\{W_1^+ \cdot \delta_1^+ + W_1^- \cdot \delta_1^- + \ldots\}$$

or

$$Min\{W_2^+ \cdot \delta_2^+ + W_2^- \cdot \delta_2^- + \ldots\}$$

(9)

Comments

As before, this case will be compiled as two separate (but related) intentions.

Inconsistent Trade-offs

When users enter pairwise trade-offs there is a chance that inconsistent relations will enter the system. For example, the user states that the slope of the trade-off between variables x and y, and between x and z are +2 and +3, respectively. Then he specifies the trade-off slope between y and z as +5. But, the slope implied through the first two trade-offs is +6. Such potential inconsistency will be handled in one of the following ways.

Forbidding inconsistent trade-offs

To prevent any chance for inconsistency we allow entering up to (n-1) distinct trade-offs (n is the cardinality of the vector x). Consider the $n \times n$ matrix of all pairwise combinations.

- Check out the n entries along the diagonal.
- Accept a new trade-off only if it refers to an entry in the matrix that is not checked. That is, stop accepting new trade-offs when all the entries in the matrix are checked.
- Each time a new trade-off is defined (say between x_i and $x_j, i \neq j$), check out both entries {ij} and {ji}. Also, for all k such that entry {ik} is already checked out, check out entries {jk} and {kj}.

Allowing inconsistent trade-offs

Here we allow up to $\frac{n \cdot (n-1)}{2}$ trade-offs. But, any new trade-off that is entered is compared to a list of implied trade-offs that are gradually being constructed as more trade-offs are recorded. If the new trade-off contradicts an implied one, the system will automatically generate a warning to the user allowing him to either accept the implied relations or stay with the inconsistent definition he has just entered.

Note that in both cases, each of the trade-offs that eventually enter the system is weighted according to the user specification but the overall weight associated with each variable is limited by the "finite capital" principle which is defined later in this document.

OR conditions (disjunct compilation)

This section describes the problem of compiling several disjunctive constraints into the GP model. This is a challenging task since all the constraints of a GP model are, by definition, implicitly conjuncts. Therefore, we need to provide a translation method for a disjunctive expression so that we can require the satisfaction of an arbitrary subset of the disjoint constraints; That is, enabling solutions to problems such as asking to satisfy exactly three disjoint constraints out of seven or asking to satisfy at least two such constraints out of five, etc. Moreover, this translation must consist of only linear expressions, as we are restricting ourselves to integer and linear-programming capabilities.

Problem description

Notation (general)

$X = \{x_1, \ldots x_k\}$ A set of *k variables.*

$f(X)$ A linear combination over the vector *X.*

z, w, b Symbols for binary variables, that is, $z, w, b \in \{0,1\}$. These variables will be subscripted as necessary.

d_i *Relation* of the form $f(X)\{\leq, \geq, <, >, =\}T_i$, participating in a disjunction.

$d_i^{\neg}$ *Complementary relation* for d_i, *e.g., if* $d_i : x_j \leq T_i$ *then* $d_i^{\neg} : x_j > T_i$.

c_i *Relation* of the form $f(X)\{\leq, \geq, <, >, =\}T_i$, participating in conjunction.

$c_i^{\neg}$ *Complementary relation* for c_i, *e.g., if* $c_i : x_j \leq T_i$ *then* $c_i^{\neg} : x_j > T_i$.

D_m^k A *disjunction* of *m* relations over the set *X* with *k* variables.

C_n^l A conjunction of *n* relations over the union of set *X* with a set of additional binary variables, and another set of constants (upper and lower bound values). The parameters *l* and *n* are defined later on.

S_q^r A predicate describing the cardinality of the subset of satisfied disjuncts in D_m^k, with a quantifier, *q*={At least, At most, Exactly}, and cardinality *r*.

The predicate has the form:

{At least, At most, exactly} r disjuncts in D_m^k *must be satisfied.*

The quantifiers {At least, At most, Exactly} will be represented by the symbols $\{\geq, \leq, =\}$, respectively.

Comments

The disjuncts and conjuncts will necessarily use different sets of variables, as the conjuncts will be introduced with additional binary variables.

Representing strong-inequalities

By *strong-inequality* we refer to a relation with the operators $\{<,>\}$, and by *weak-inequality* we refer to a relation with the operators $\{\leq,\geq\}$.

Notice that in a LP package we do not have the capability to express strong-inequalities such as $x<7$ or $x>12$, due to the representation of LP models in those packages. Therefore, we represent strong-inequalities using weak inequalities where the targets on the RHS are perturbed by an infinitesimal amount (ε).

I.e., $f(X)<T$, is represented by: $f(X)\leq T-\varepsilon$, and,
(10)
$f(X)>T$, is represented by: $f(X)\geq T+\varepsilon$.
(11)

Although this solution seems logically incorrect, it is practically sound, since a LP package uses such an ε anyway, e.g., in order to round values.

Problem description

Given a set of disjuncts, $D_m^k=\{d_1\ldots d_m\}$ *over X, a requirement* S_q^r and a set of bounds for all the variables: $L_j\leq x_j\leq U_j, \forall x_j\in X$, $1\leq j\leq k$.

Our objective is to determine C_n^l so that it satisfies the requirements S_q^r over D_m^k.

The solution of the problem will be presented as follows:

1. First we introduce a solution for the general case D_m^k, S_q^r, showing how we represent inequality disjuncts as conjuncts, and then how to expand this representation to include equality disjuncts.

We also show a solution for the special case when $k=1$ (that is, for a single variable), and all disjuncts are equality relations.

Comments

1. The requirement of an upper/lower-bound for x_j is mandatory for the solution to work, as these bounds are used in the re-formulation of the problem.

Given the upper and lower bounds of the decision variables, we can calculate upper and lower bounds of any function $f(X)$ over these variables. See 0.

Alternative solutions

General case solution

In the general solution we first show how to solve inequality relations, then handle equality relations, by representing every equality relation by two inequalities.

Representation of inequalities

First, we notice that an inequality of the general form $d: f(X)\{\le,\ge\}T$, may be re-represented as follows:

1. A relation of the form $d_i : f_i(X) \le T_i$ by:

$$c_i : f_i(X) \le z_i \cdot T_i + (1 - z_i) \cdot U(f_i)$$

(12a)

If $z = 1$, we get $f_i(X) \le T_i$, thus, we require that d_i must be satisfied.

If $z = 0$, we get $f_i(X) \le U(f_i)$, thus, making it a redundant constraint, i.e., one that is trivially satisfied.

The complement $d_i^{\neg} : f_i(X) > T$ is represented by:

$$c_i^{\neg} : f_i(X) \ge (1 - z_i) \cdot (T_i + \varepsilon) + z_i \cdot L(f_i)$$

(12b)

If $z = 1$, we get $f_i(X) \ge L(f_i)$, thus, making it a redundant constraint.

If $z = 0$, we get $f_i(X) \ge (T_i + \varepsilon)$, thus, we require that $d_i^{\neg}$ must be satisfied.

Notice that the conjuncts c_i and $c_i^{\neg}$ together, are to be satisfied at the same time (they are conjuncts), thus: If z=1, d_i is satisfied, and if z=0, d_i is not satisfied.

2. A relation of the form $d_i : f_i(X) \geq T_i$, is represented respectively, as described above:

$$c_i : f_i(X) \geq z_i \cdot T_i + (1 - z_i) \cdot L(f_i)$$

$$c_i^{\neg} : f_i(X) < (1 - z_i) \cdot (T_i - \varepsilon) + z_i \cdot U(f_i)$$

(13)

Comments

1. The representation above keeps the relations linear.
2. For any value of z_i, either d_i or $d_i^{\neg}$ will be satisfied while the other will not.

The presentation of *U(f)* and *L(f)* is made possible due to the requirement that all variables are bounded.

Representation of Equalities

An equality relation $d_i : f_i(X) = T_i$ can be rewritten using two inequalities as follows: $(f_i(X) \leq T_i) \wedge (f_i(X) \geq T_i)$.

We use two binary variables, z_i for representing $(f_i(X) \leq T_i)$ and w_i for representing $(f_i(X) \geq T_i)$, such that d_i is satisfied *iff* $z_i = w_i = 1$.

Using the inequality representations from the previous section we get:

$$c_{zi}: f_i(X) \leq z_i \cdot T_i + (1 - z_i) \cdot U(f_i)$$
$$c_{zi}^{\neg}: f_i(X) \geq (1 - z_i) \cdot (T_i + \varepsilon) + z_i \cdot L(f_i)$$
$$c_{wi}: f_i(X) \geq w_i \cdot T_i + (1 - w_i) \cdot L(f_i)$$
$$c_{wi}^{\neg}: f_i(X) \leq (1 - w_i) \cdot (T_i - \varepsilon) + w_i \cdot U(f_i)$$
$$c^{wz}: z_i + w_i \geq 1$$

(14)

Satisfying d_i:

If $z_i = 1, w_i = 1$, we get $(f_i(X) \geq T_i)$ *and* $(f_i(X) \leq T_i)$, thus, $(f_i(X) = T_i)$.

Unsatisfying d_i:

If $z_i = 1, w_i = 0$, we get $(f_i(X) \leq T_i - \varepsilon)$, especially $(f_i(X) \neq T_i)$.

If $z_i = 0, w_i = 1$, we get $(f_i(X) \geq T_i + \varepsilon)$, especially $(f_i(X) \neq T_i)$.

Comments

1. We eliminate the case $z_i = 0, w_i = 0$, as it leads to an inconsistency when requiring $(f_i(X) \geq T_i + \varepsilon)$ *and* $(f_i(X) \leq T_i - \varepsilon)$, simultaneously.

The formulation above has no preference for either w_i or z_i, so that if d_i should not be satisfied, $f_i(X) \neq T_i$, we can choose either direction $f_i(X) < T_i$ or $f_i(X) > T_i$ according to the values of X.

Problem description

Given the set D_m^k of m disjunct relations and the requirement S_q^r; and given that D_m^k consists of m_1 inequalities and m_2 equalities $(m_1 + m_2 = m)$, and bounds for the variables, $L_j \leq x_j \leq U_j, \forall x_j \in X$, $1 \leq j \leq k$.

We present the set C_n^l of $n = 2 \cdot m_1 + 5 \cdot m_2 + 1$ conjuncts, and $l = m_1 + 2 \cdot m_2 + k$ variables, and show that C_n^l is satisfied *iff* D_m^k, S_q^r is satisfied.

Construction of C_n^l

As we showed above, an inequality relation d_i can be presented by two conjuncts c_i and $c_i^{\neg}$ and a binary variable z_i. An equality relation d_i can be represented by five conjuncts c_i^{z}, $c_i^{z\neg}$, c_i^{w}, $c_i^{w\neg}$ and c^{wz}, and two binary variables z_i and w_i.

Given m_1 inequalities and m_2 equalities we construct C_n^l, by introducing $m_1 + 2 \cdot m_2$ new binary variables, $z_i \in \{0,1\}, 1 \le i \le m$, and $w_j \in \{0,1\}$, $1 \le j \le m_2$. (*Notice,* $m_1 + 2 \cdot m_2 = m + m_2$)

$$\begin{aligned} C_n^l = \quad & \left\{c_1^{z} \cdots c_{m2}^{z}, c_1^{z\neg} \cdots c_{m2}^{z\neg}, c_1^{w} \cdots c_{m2}^{w}, c_1^{w\neg} \cdots c_{m2}^{w\neg}, c_1^{wz} \cdots c_{m2}^{wz}\right\} \\ & \cup \left\{c_{m2+1} \cdots c_m, c_{m2+1}^{\neg} \cdots c_m^{\neg}\right\} \\ & \cup \left\{c^{z}\right\} \end{aligned}$$

(15)

Where the representation of c^z depends on the type of relations as follows: for $1 \le i \le m$, $1 \le j \le m_2$

$S_{\ge}^r \qquad c^z : \sum_i z_i + \sum_j w_j \ge r + m_2$

$S_{\le}^r \qquad c^z : \sum_i z_i + \sum_j w_j \le r + m_2$

$S_{=}^r \qquad c^z : \sum_i z_i + \sum_j w_j = r + m_2$

(16)

Comments

1. The $2 \cdot m_1 + 5 \cdot m_2 + 1$ conjuncts are: $2 \cdot m_1$ conjuncts for handling the inequality relations, and $5 \cdot m_2$ conjuncts for handling the equality relations, and an additional requirement to satisfy S_q^r.

The $m_1 + 2 \cdot m_2 + k$ variables are: m_1 binary variables for the inequality relations, and $2 \cdot m_2$ binary variables for the equality relations, and the k variables in X

2. Note that all the expressions in (15) and (16) are linear.

3. The conjuncts $\left\{c_1^{z} \cdots c_{m2}^{z}, c_1^{z\neg} \cdots c_{m2}^{z\neg}, c_1^{w} \cdots c_{m2}^{w}, c_1^{w\neg} \cdots c_{m2}^{w\neg}, c_1^{wz} \cdots c_{m2}^{wz}\right\}$ represent m_2 equality relations according to (14) above.

Each equality relation d_i is represented in this set by $\left\{c_i^{z}, c_i^{z\neg}, c_i^{w}, c_i^{w\neg}, c_i^{wz}\right\}$, and will be satisfied *iff* $w_i = z_i = 1$ (i.e., when $w_i + z_i = 2$); and therefore, it will not be satisfied when $w_i + z_i = 1$.

4. The conjuncts $\left\{c_{m2+1} \cdots c_m, c_{m2+1}^{\neg} \cdots c_m^{\neg}\right\}$ represent m_1 inequalities according to (12), for an inequality with the $\leq$ operator, or (13) for an inequality the $\geq$ operator.

Each inequality relation d_i will be satisfied *iff* $z_i = 1$; therefore, it will not be satisfied when $z_i = 0$.

5. The conjunct c^z described in (16) represents the requirement of S_n^r:

i. $S_=^r$ $\quad c^z : \sum_i z_i + \sum_j w_j = r + m_2$

Notice that all the variables in the equation are binary variables, that it, they may only add one or zero to the sum.

An inequality will be satisfied *iff* its binary variable z_i adds one to the sum, or adds zero otherwise.
An equality relation will be satisfied *iff* its binary variables z_i and w_i add two to the sum. Otherwise they add one.

Thus, in order to satisfy any *r* conjuncts or disjuncts we require:

$$S^{r}_{=} \qquad c^{z}: \sum_{inequalities} z_i + \sum_{equalities} \left(z_i + w_j - 1\right) = r$$

However, there are m_2 equalities, providing ($-m_2$) to the total sum, thus, we get the equation as presented in (16).

ii. $S^{r}_{\le(\ge)} \qquad c^{z}: \sum_{i} z_i + \sum_{j} w_j \le (\ge)\ r + m_2$

According to the explanation above, if we wish to satisfy at least (at most) *r* disjuncts in D^{k}_{m}, the result for the At least (respectively, At most), case is immediate, as the sum represents exactly how many disjuncts are satisfied, while all the other disjuncts will not be satisfied.

Example

$D_4^1 : (x = 9)^I \vee (x = 13)^{II} \vee (x \geq 17)^{III} \vee (x \leq 8)^{IV},\ S_=^1$
$(k = 1,\ \ m = 4,\ m1 = 2,\ m2 = 2)$

$C_{15}^{\cdot 7}$:
$c_1^z \qquad x \leq z_1 \cdot 9 + (1 - z_1) \cdot U^x \qquad (I)$
$c_1^{z\neg} \qquad x \geq (1 - z_1) \cdot (9 + \varepsilon) + z_1 \cdot L^x$
$c_1^w \qquad x \geq w_1 \cdot (9) + (1 - w_1) \cdot L^x$
$c_1^{w\neg} \qquad x \leq (1 - w_1) \cdot (9 - \varepsilon) + w_1 \cdot U^x$
$c_1^{wz} \qquad w_1 + z_1 \geq 1$
$\wedge$
$c_2^z \qquad x \leq z_2 \cdot 13 + (1 - z_2) \cdot U^x \qquad (II)$
$c_2^{z\neg} \qquad x \geq (1 - z_2) \cdot (13 + \varepsilon) + z_2 \cdot L^x$
$c_2^w \qquad x \geq w_2 \cdot (13) + (1 - w_2) \cdot L^x$
$c_2^{w\neg} \qquad x \leq (1 - w_2) \cdot (13 - \varepsilon) + w_2 \cdot U^x$
$c_2^{wz} \qquad w_2 + z_2 \geq 1$
$\wedge$
$c_3 \qquad x \geq z_3 \cdot 17 + (1 - z_3) \cdot L^x \qquad (III)$
$c_3^{\neg} \qquad x \leq (1 - z_3) \cdot (17 - \varepsilon) + z_3 \cdot U^x$
$\wedge$
$c_4 \qquad x \leq z_4 \cdot 8 + (1 - z_4) \cdot U^x \qquad (IV)$
$c_4^{\neg} \qquad x \geq (1 - z_4) \cdot (8 + \varepsilon) + z_4 \cdot L^x$
$\wedge$
$(z_1 + w_1 - 1) + (z_2 + w_2 - 1) + z_3 + z_4 = 1 \qquad (S_=^1)$
$c^z \qquad \equiv z_1 + z_2 + z_3 + z_4 + w_1 + w_2 = +3$

$s.t.\ z_1, z_2, z_3, z_4, w_1, w_2 \in \{0,1\},$
$L^x \leq x \leq U^x$

Handling equalities (a single variable)

Compiling D_m^k is an easier task, in the special case where $k = 1$ (a single variable) and all the disjuncts are equality relations, with the requirement $S_=^1$ (that is exactly one disjunct is required to be satisfied).

Another required restriction is that all the equalities in D_m^1 are unique, that is:

For $d_i : x = T_i$ and $d_j : x = T_j, i \neq j$ implies that $T_i \neq T_j$.

Constructing C_2^{m+1}

Given the set D_m^1 of m equality disjuncts and $S_=^1$, we show how to generate a conjunction C_2^{m+1}, with two conjuncts and $m+1$ variables, which is satisfied *iff* D_m^1, S_q^r are satisfied.

We attach to every $d_i : x = T_i$ a binary variable b_i, as a flag, to the value T_i; then request, according to S_q^r, that {At least, At most, Exactly} r flags will be satisfied.

Constructing C_2^{m+1}, by introducing m new binary variables, $b_i \in \{0,1\}, 1 \leq i \leq m$:

$$C_2^{m+1} = \{c_1\} \cup \{c^b\}$$

(17)

The solution is immediate:

$$c_1 : x = b_1 \cdot T_1 + \ldots + b_m \cdot T_m$$

(18)

$$c^b : \sum_i b_i = 1$$

(19)

Comments

1. The requirement set S_{-}^{1} aims to eliminate redundancy. Moreover, if we allow any requirement set S_q^r then a subset of disjuncts satisfying S, must be similar; that is all of them with the form $x = T$.

2. Notice that C_2^{m+1} is represented by two groups; the first one $\{c_1\}$ is the translation of D_m^1 constraints from disjuncts into conjuncts; and the second $\{c^h\}$ is the requirement for satisfaction presented by S_q^r.

3. Note that the expressions (18) and (19) are linear.

4. First we notice according to (19) that only (and exactly) one of the binary variables will have the value one, whereas all the other binary variables will have the value zero.

That is, for some $1 \le i \le m$, $b_i = 1$, and $b_j = 0$ $1 \le j \le m$, $j \ne i$.

5. According to the above, c_1 in (18) will always have the form $x = T_i$ $(b_i = 1)$.

Therefore, the above formulation allows us to *choose* exactly one of the values t_i to be assigned to x.

Related issues

See section 0 about discrete variable compilations.

Example

$$D_4^1 : \{(x = 4) \vee (x = 40) \vee (x = -10) \vee (x = 12)\}, \quad S^1$$

C_2^5 :

$$c_1 : \quad x = b_1 \cdot 4 + b_2 \cdot 40 - b_3 \cdot 10 + b_4 \cdot 12$$

$\wedge$

$$c^b : \quad b_1 + b_2 + b_3 + b_4 = 1$$

$$s.t. \; b_1, b_2, b_3, b_4 \in \{0,1\}$$

Discrete variables compilation

In the first two sections we described ordinary goals and trade-offs. However, those sections dealt with continuous variables only. This section deals with the problem of representing discrete-valued variables. Especially, we are interested in representing string values, such as the names of hotels in a city, or colors of the product, etc.

Related issues

1. See section 0 about Combining objectives.

Representation of strings as numerical values is out of the scope of this section.

The general case

Input for compilation

(Regarding a variable t)

Feasible values of t $\{T_1 \ldots T_m\}$.

Level of objective function L.

Preference weights for every feasible value of t $\{W_1 \ldots W_m\}$.

Relative importance within the level, V_t.

Representation of variables and constraints

Definition of t: $t = \sum_{i=1}^{m} T_i \cdot b_i$

The definition of t is kept OUTSIDE of the GP. We use it only for the purpose of translating the GP outputs in order to present them to the user (or to forward them to the agent of the other party). The other variables we define below will play a role in the GP.

Hard constraints: $x_t = \sum_{i=1}^{m} W_i \cdot b_i$ x_t is an auxiliary variable.

$$\sum b_i = 1,\ \ b_i \in \{0,1\},\ 1 \le i \le m$$

$0 \le c_i \le 1,\ 1 \le i \le m$ c_i are continuous variables that serve as a relaxation for the binary variables b_i.

Goal constraint: $x_i / W^{\max} + \delta^- - \delta^+ = 1$

$W^{\max}$ is the maximal weight in $\{W_1 \ldots W_m\}$.

(20)

Objective (at level L*): Min* $Z = V_t \cdot \delta^- + \ldots$

Comments

1. The goal constraint described above is a one-sided one, where $\delta^{+} = 0$.

2. The representation above preserves our requirement that the objective function can consist of deviation variables only.

3. *UI Issues:* The definition of variable t is not within the GP problem. That is, the GP problem will deal only with the *other* variables we define above.

4. The variable x is auxiliary, thus, we must also set its upper/lower bounds. These may simply be $x \in [0, W^{\max}]$

Example 1

Suppose that the variable t represents the day of delivery within a week, where the days Sunday through Saturday, are represented using the values one to seven, respectively.

A seller can deliver the product on Tuesdays, Wednesdays, and Fridays, with the following preference weights: $\{W_{Tue} = 1, W_{Wed} = 8, W_{Fri} = 3\}$.

Also suppose that the seller asked for some Level of objective L, and some relative importance within that level V_t, and that the seller cannot provide the product on any of the other days.

Definition of t (day): $t = 3 \cdot b_1 + 4 \cdot b_2 + 6 \cdot b_3$

Definition of x: $x = 1 \cdot b_1 + 8 \cdot b_2 + 3 \cdot b_3$

$$b_1 + b_2 + b_3 = 1, \quad b_1, b_2, b_3 \in \{0,1\}, \quad 0 \le x \le 8$$

Goal constraint: $x/8 + \delta^{-} - \delta^{+} = 1$

Objective (level L): $Min \quad Z = V_t.\delta^-$

Preliminary Negotiation Stage

Since the x_i variables participate in negotiation phases in which the parties may attempt to adjust them in small or moderate steps, we decompose the negotiations into a preliminary stage in which the binary variables b_i are relaxed through c_i. This stage ends with an attempt by the party who is ready to accept an offer made by the other side to re-adjust the x_i values according to the binary (b_i) rather than the continuous variables (c_i). This is illustrated through the following example.

Example 2

Suppose there are five colors, denoted by the index i = 1, ..., 5 and weighted by the two parties as given below.

Color	Red	Orange	Yellow	Green	Red
Index (i)	1	2	3	4	5
Buyer's Weights (W_B)	5	3	3	2	1
Seller's W Weights (W_S)	2	3	4	3	4

The part in the Buyer's original GP that is relevant to our example is:

$$\alpha = 5 \cdot \delta_B^-$$
$$X_B = 5 \cdot b_1 + 3 \cdot b_2 + 3 \cdot b_3 + 2 \cdot b_4 + 1 \cdot b_5$$
$$b_1 + b_2 + b_3 + b_4 + b_5 = 1$$
$$\frac{X_B}{5} + \delta_B^- - \delta_B^+ = 1$$
$$b_i \in \{0,1\} \qquad \forall i$$

The corresponding part in the Seller's GP is:

$$\beta = 4 \cdot \delta_S^-$$
$$X_S = 2 \cdot b_1 + 3 \cdot b_2 + 4 \cdot b_3 + 3 \cdot b_4 + 4 \cdot b_5$$
$$b_1 + b_2 + b_3 + b_4 + b_5 = 1$$
$$\frac{X_S}{4} + \delta_S^- - \delta_S^+ = 1$$
$$b_i \in \{0,1\} \qquad \forall i$$
$$X_S \geq 3$$

Notice that since the seller's highest ranked alternative was accorded a score of 4 his GP is trying to assign that value to X_S while in the buyer's case the top choice was given a score of 5 and therefore his GP tries to set $X_B = 5$.

To generate the Seller's initial offer we employ the original binary variables (the b_i's). The outcome is:

$$b_5 = 1, b_1 = b_2 = b_3 = b_4 = 0, \quad X_S = 4, \delta_S^- = 0 \qquad \Rightarrow \beta = 0$$

$$\Rightarrow X_B = 1, \delta_B^- = 0.8 \qquad \Rightarrow \alpha = 4$$

The Seller's second offer is dictated by the Aggressive-Improve model (say, $\rho = 0.1$). In generating this (and all subsequent) offer(s) we replace the b_i's with their continuous counterparts (the c_i's). The outcome is:

$$\alpha^{new} + \gamma^- - \gamma^+ = \alpha^{last} - \rho \cdot \left(\alpha^{last} - \alpha^*\right) \Rightarrow \alpha^{new} + \gamma^- - \gamma^+ = 4 - 0.1 \cdot (4 - 0) = 3.6$$

$$\Rightarrow \delta_B^- = 0.72 \Rightarrow X_B = 1.4 \Rightarrow c_1 = 0.6, c_2 = 0.4 \Rightarrow X_S = 3.4 \geq 3$$

The Seller's next offer, again dictated by the Improve model is:

$$\alpha^{new} + \gamma^- - \gamma^+ = \alpha^{last} - \rho \cdot \left(\alpha^{last} - \alpha^*\right) \Rightarrow \alpha^{new} + \gamma^- - \gamma^+ = 3.6 - 0.1 \cdot (3.6 - 0) = 3.24$$

$$\Rightarrow \delta_B^- = 0.648 \Rightarrow X_B = 1.76 \Rightarrow c_1 = 0.24, c_2 = 0.76 \Rightarrow X_S = 3.24 \geq 3$$

This will continue in the same manner until we reach acceptance. At that point do wish to translate the c_i's back into binary values for the corresponding b_i's so as to select a unique color. Suppose the buyer is ready to accept the last seller's offer (with the values given above). To do so, the buyer will solve an extended GP in which the $\alpha^{last} = 3.24$, $\beta^{last} = 1.76$ values given in the last seller's offer serve as targets:

$$\begin{array}{ll}
Min \quad 1000\cdot\Delta_{\beta}^{+}+100\cdot\Delta_{\alpha}^{+}+10\cdot\left(\tau^{-}+\tau^{+}\right)+\left(\upsilon^{-}+\upsilon^{+}\right) & \\
s.t. & \\
\alpha+\Delta_{\alpha}^{-}-\Delta_{\alpha}^{+}=3.24 & \beta+\Delta_{\beta}^{-}-\Delta_{\beta}^{+}=1.76 \\
X_{B}+\upsilon^{-}-\upsilon^{+}=1.76 & X_{S}+\tau^{-}-\tau^{+}=3.24 \\
X_{B}-\sum_{i} W_{Bi}\cdot b_{i}=0 & X_{S}-\sum_{i} W_{Si}\cdot b_{i}=0 \\
\qquad\qquad b_{i}\in\{0,1\} & \forall i \\
GP_{Buyer} & GP_{Seller}
\end{array}$$

If this attempt to finalize the deal fails (e.g., due to the hard constraint embedded in the GP of either the buyer or the seller) we let the parties continue negotiate. If such failures repeat themselves we can still try to resolve the situation through the mechanism level (for example, switching into an Ignorant mode). This will be defined in the Mechanism document.

Compilation of Date

Dates are represented as continuous variables that count the minutes from the beginning of 1.1.1970. To compile date-type variables we translate the absolute date into a relative date. The GP will handle only the relative date variable. To do so, we refer to the lower bound of the absolute date variable as Offset_D (this is the number of minutes from 1.1.1970 to the lower bound). Then, we do the following translations:

The absolute target date specified by the user for D (Abs_tar_D) will be expressed in relative terms as: Rel_tar_D = Abs_tar_D – Offset_D.

The relevant goal constraint will be: $D+\delta_{D}^{-}-\delta_{D}^{+}=\mathrm{Re}l\_tar\_D$.

The optimal value (D*) will be translated back to absolute date terminology through: Abs_D* = Offset_D + D*.

The trivial case

Input for compilation

(Regarding a variable t)

Feasible values of t $\{T_1 \ldots T_m\}$,

Level of objective function L.

Relative importance within level V_t.

The values of t are ordered, i.e., $t \in \{T_1, \ldots, T_1 + m - 1\}$. Moreover, the order of the values represents the user order of preference for these values. I.e. the weights for these values are implicitly $\{W_1 = 1, \ldots, W_m = m\}$.

Representation in the GP problem

In the described simple case, we set the variable t to be of an Integer type, so that it can be assigned only with integral values. Then we represent the goal as an ordinary goal (see section 0 about Ordinary goals)

Definition of t: $t \in [T_1, \ldots, T_1 + m - 1]$

I.e. t is *Integer* and T_1, $T_m = T_1 + m - 1$, are the lower/upper bounds for t.

Goal constraint: $t/T_m + \delta^- - \delta^+ = 1$

Objective (at level L): *Min* $Z = V_t \cdot \delta^- + \ldots$

(21)

Comments

1. The definition of variables t and x as presented in (20) above will be identical in this case. Therefore, we omit the usage of the auxiliary variable x.

2. Notice that in this case we do not need to represent t explicitly with binary flags.

3. *UI issues:* with suitable UI support, we can easily allow any value T_j, $1 \le j \le m$, to be the most preferred value (instead of T_m), and turn the goal constraint into two-sided one, with deviation weights W_t^-, W_t^+ as with ordinary goals.

Example

Suppose that the variable t represents the day of delivery within a week, where the days Sunday through Saturday, are represented using the values one to seven respectively.

A seller can deliver the product on Mondays, Tuesday, and Wednesdays, in this order of preference, i.e., Wed. is the most preferred.

Also suppose that the user asked for some Level of objective L, and some Relative importance within that level V_l, and that the seller cannot provide the product on Sundays.

Definition of t: $t \in [2..4]$, *t is integer.*

Goal constraint: $t/4 + \delta^- - \delta^+ = 1$

Objective (level L): *Min* $Z = V_t \cdot \delta^- + \ldots$ Combining objectives

The "finite capital" problem

This section explains how to weigh the deviations associated with a particular decision variable within a specific value of the lexicographic objective function. We assume that the variable will appear in the goal constraints in two ways: single-variable and two-variable (trade-off) goal constraints.

Single-variable goal constraints: $x_i/g_i + \delta_i^- - \delta_i^+ = 1$

Two-variable goal constraints: $x_i/a_{ij} - b_{ij} \cdot x_j/a_{ij} + \gamma_{ij}^- - \gamma_{ij}^+ = 1$

Weights are to be elicited in the following manner:

1. Each variable is associated with a particular level of the objective function

2. Within its level, each variable is assigned a weight (V_i) that determines its relative importance vis-à-vis the other variables in that level.

3. For each variable we enable the user to define the relative importance of positive vs. negative deviations (W_i^+, W_i^-). Three scenarios might be realized:

   i. $W_i^+ > W_i^- > 0$ or its symmetric case $W_i^- > W_i^+ > 0$.

ii. $W_i^+ > 0,\ W_i^- = 0$ or its symmetric case $W_i^- > 0,\ W_i^+ = 0$.

iii. $W_i^+ > 0,\ W_i^- < 0$ or its symmetric case $W_i^- > 0,\ W_i^+ < 0$.

Trade-offs are allowed only between pairs of variables associated with the same objective level. The weight of each trade-off goal is either provided explicitly from the GUI layer, or computed as an arithmetic average of the values that are involved in it (i.e., $V_{ij} = \frac{V_i + V_j}{2}$). With respect to deviations above or below the trade-off line we consider only two scenarios:

a. $W_{ij}^+ = W_{ij}^- > 0$

b. $W_{ij}^+ = 0,\ W_{ij}^- > 0$ or its symmetric case $W_{ij}^- = 0,\ W_{ij}^+ > 0$

The *finite capital* problem is how to ensure that the importance of a particular variable in a given level will not be distorted as a result of its participation in many (or few) trade-off goal constraints. Let T(i) represent the set of variables that are traded off against variable i. We distinguish between two GUI-dependent cases:

1. The trade-off constraint was entered as a linear function where one of the variables (x_i at the top part of this page) is expressed in terms of the second (x_j). In that case, the deviations are measured with respect to x_i and therefore the deviation variables that correspond to this trade-off will affect only the portion related to x_i in the objective function. In this case, the part of the objective function that corresponds to x_i will be formulated as follows:

$$Min \quad V_i \cdot \left[\frac{V_i}{V_i + \sum_{j \in T(i)} V_{ij}} \cdot \left(\frac{W_i^- \cdot \delta_i^- + W_i^+ \cdot \delta_i^+}{W_i^- + W_i^+} \right) + \sum_{j \in T(i)} \frac{V_{ij}}{V_i + \sum_{j \in T(i)} V_{ij}} \cdot \left(\frac{W_{ij}^- \cdot \gamma_{ij}^- + W_{ij}^+ \cdot \gamma_{ij}^+}{W_{ij}^- + W_{ij}^+} \right) \right]$$

2. The trade-off constraint was entered through two points that represent equivalent benefit to the user. In that case the constraint should affect both x_i and x_j. In this case, the part of the objective function that corresponds to x_j will include a

term that corresponds to the same trade-off that entered the part that corresponded to x_i, but with orthogonal deviation variables (notice that, $\gamma_{ji}^{+} = b_{ij} \cdot \gamma_{ij}^{-}, \gamma_{ji}^{-} = b_{ij} \cdot \gamma_{ij}^{+}$).

Comments

1. In the formula above the first term corresponds to "capital allocation" to the ordinary non-trade-off objective, the second term corresponds to the trade-off terms.

2. If there are no trade-off constraints associated with variable i then only the first term appears and the weighted sum of the upper and lower deviations of that variables are accorded the proper weight of V_i.

3. As the number of trade-off constraints associated with variable i increases, the relative importance of the first term (corresponding to the single-variable constraint) decreases.

A general compilation example

An agent who sells used cars is willing to sell a car with the following requirements

Price: represented by variable p. (See 0.)

Price ranges between [2200-3500] in $US.

The target price is $T_p = 3000$.

The Relative importance for price is $V_p = 1000$.

Also assume equal preference for deviation in either direction, i.e. $W_p^- = W_p^+ = 1.0$.

Delivery day: represented by variable d. (See 0.)

Delivery day ranges between [10-18] from the day of signing the deal.

The target day is $T_d = 12$.

The relative importance for delivery is $V_d = 100$.

Also assume the following preference for deviation in either direction: $W_d^- = 0.666, W_d^+ = 1.0$.

Color: represented by the variable t. (See 0)

Color is one of the following: Red, Black, and Silver.

Color strings have the translation values $t \in \{17, 22, 50\}$ respectively.

The relative importance for color is $V_d = 10$.

The colors' weights are $W = \{10, 20, 30\}$, respectively. For example, these weights may have some relevance to the number of cars that remained in store for some color).

All the variables are at the same level in the objective function.

Trade-off: here the trade-off is between the price and the delivery day, e.g., for better usage of the parking space available to the seller. (See 0).

Suppose that the seller provided the following two points for the trade-off, $\{12, 3000\}$ representing the respective goals' targets, and point $\{15, 3090\}$, i.e., representing a loss of $30 for every day of delay in delivery.

The relative importance of the trade-off is $V_{dp} = 300$.

Also assume equal preference for deviation in either direction, i.e. $W_{dp}^- = W_{dp}^+ = 1.0$.

Comments

1. Notice that the point $\{12, 3000\}$ represents the targets of the *ordinary* goals for the two variables participating in the trade-off. In general, however, such points do not necessarily reside on the two-dimensional trade-off straight line.

2. Since the trade-off was entered through two points it will affect both the part of the objective that deals with the price and the part that deals with delivery. On the other hand, if the trade-off was entered as a linear function where price is expressed through delivery, the corresponding deviations would have appeared only in the part that corresponds to price.

$$\begin{aligned} Z = Min \quad & 1000 \cdot \left[\frac{1000}{1300} \cdot \left(\delta_p^- + \delta_p^+ \right) + \frac{300}{1300} \cdot \left(\delta_{pd}^- + \delta_{pd}^+ \right) \right] + \\ & 100 \cdot \left[\frac{100}{400} \left(0.666 \cdot \delta_d^- + \delta_d^+ \right) + \frac{300}{400} \cdot \left(\tfrac{30}{2} \cdot \delta_{pd}^- + \tfrac{30}{2} \cdot \delta_{pd}^+ \right) \right] + \\ & 10 \cdot \delta_x^- \end{aligned}$$

subject to:

$$\begin{aligned} & p/3000 + \delta_p^- - \delta_p^+ = 1 && (price) \\ & d/12 + \delta_y^- - \delta_y^+ = 1 && (day) \\ & x/30 + \delta_x^- - \delta_x^+ = 1 && (color) \\ & p/2640 - 30/2640 \cdot d + \delta_{pd}^- - \delta_{pd}^+ = 1 && (price - day\,tradeoff) \end{aligned}$$

$$\begin{aligned} & t = b_1 \cdot 17 + b_2 \cdot 22 + b_3 \cdot 50 && (representation\ of\ t) \\ & x = b_1 \cdot 10 + b_2 \cdot 20 + b_3 \cdot 30 && \\ & b_1 + b_2 + b_3 = 1 && \end{aligned}$$

$$\begin{aligned} & 2200 \le p \le 3500 && (bounds\ from\ UI) \\ & 10 \le d \le 18 && \end{aligned}$$

$$\begin{aligned} & 0 \le \delta_p^- \le 800/3000 && (compiled\ bounds) \\ & 0 \le \delta_p^+ \le 500/3000 && \\ & 0 \le \delta_d^- \le 2/12 && \\ & 0 \le \delta_d^+ \le 6/12 && \\ & 0 \le x \le 30 && \\ & 0 \le \delta_x^- \le 1 && \\ & 0 \le \delta_x^+ \le 0 && \\ & 0 \le \delta_{dp}^- \le 3180/2640 && \\ & 0 \le \delta_{dp}^+ \le 3180/2640 && \\ & b_1, b_2, b_3 \in \{0,1\} && \end{aligned}$$

Now, assume that the relative importance of the trade-off was not provided explicitly as above, and is therefore derived as the average of the values of V_i and V_j. The resulting GP is identical to the previous one, except that V_{ij} =300 is now replaced with V_{ij} =550 (and consequently, the values 1300 and 400 in the denominators are replaced by 1550 and 650, respectively).

Introduction

This document describes the *utility layer* of an *automatic negotiation system*. An automatic negotiation system offers facilities for automated negotiations that may be used by a variety of applications in eCommerce, Communication, Financial Services, Insurance and more. The utility layer provides means for handling offers. Specifically, it includes facilities for suggesting, verifying, completing, evaluating, ranking, modifying and updating offers. Here, offers may be multi-item, multi-attribute complex offers with various constraints, preferences and trade-offs. The underlying assumption is that such constraints, preferences and trade-offs are expressed in terms of *Goal Programs* (GP). Goal programs are well known in the scientific literature and their usage as a foundation for negotiations was first described in PCT/IL00/00516.

To summarize:

- We outline a collection of manipulation procedures. This defines an *API* (application program interface) for negotiations. The API is described in terms of Goal Programs. However, it is generic in concept and can apply to other formalisms for describing a user's set of constraints, preferences and trade-offs, e.g. the Analytic Hierarchy Process (AHP), see Saaty (1980).
- We define the actual manipulation algorithms in terms of Goal Programs. These algorithms were implemented in the programming language C and tested in conjunction with publicly available software for linear and integer programming.

- Although we use the names "Bob" and "Sue" for the negotiating agents, the procedures may apply to arbitrary parties, buyers, sellers and also symmetric ones (e.g., bartering).

Notation

In describing negotiating parties we use "Bob" and "Sue" to name the parties. Each party could be a trader (buyer, seller, barterer) or more generally a parry negotiating an issue (e.g., an environmental project decision attributes).

x this is a vector of decision variables.

X this is a vector of values for decision variables.

d this is a vector of deviation variables of a goal program.

D this is a vector of values for deviation variables of a goal program.

$[X|D]$ this is a combined presentation of X and D above.

□ this is a vector of priority levels in the objective function of Bob. Here, the levels are of the form $\{\alpha_1 \ldots \alpha_k\}$ where α_i is an objective expression (min or max). We may sometimes blur the distinction between the expressions and their resulting evaluation within a particular assignment of values to deviation and decision variables.

F_α Set of goal constraints that link the elements of x with their respective deviation variables in Bob's program. By writing $x \in F_\alpha$ we loosely mean that x and the associated deviation variables satisfy this set of constraints.

S_B Set of hard constraints in Bob's program, including level-by-level.

□ Vector of priority levels in the objective function of the Sue.

F_β Set of goal constraints that link the elements of x with their respective deviation variables in the Sue's program.

S_S Set of hard constraints in the Sue's program, including level-by-level.

P_{agent}^{type}	A proposal of type = {"new", "last" or "best"} made by the agent = {"Bob" or "Sue"} (where "best" means that this proposal contains the best values offered so far from the point of view of the *other* agent). A proposal consists of: 1. A *tuple* $T = [D \mid X]$ of values for the deviation, D, and decision variables, X. 2. A *tuple* α of the objective-functions values.
x_{agent}^{type}	Values of the decision variables associated with P_{agent}^{type}.
$\alpha_{agent}^{type}, \beta_{agent}^{type}$	Values for the priority levels of Bob and Sue respectively associated with P_{agent}^{type}.
α^*, β^*	Best possible values for the priority levels of Bob and Sue respectively.
α_*, β_*	Worst possible values for the priority levels of Bob and Sue respectively.
ρ_{SS}	Proportion of improvement in Sue's utility (□) with respect to β_S^{last}.
ρ_{SB}	Proportion of improvement in Sue's utility (□) with respect to β_B^{last}.
ρ_{BB}	Proportion of improvement in Bob's utility (□) with respect to α_B^{last}.
ρ_{BS}	Proportion of improvement in Bob's utility (□) with respect to α_S^{last}.
ρ_S, ρ_B	Proportion of decrease in Sue's (Bob's) utility □ (□).
R_S	Region of acceptable □ values for Sue.
R_B	Region of acceptable □ values for Sue.
W, V	Vectors of weights on deviation variables

Notes

- A *Linear Program (LP)* problem consists of a list of constraints, a list of bounded variables, and an objective function to be minimized.

To solve a LP we currently may use any LP package (e.g., lp_solver3.0), which implements the Simplex algorithm and can handle integer variables, or other relevant algorithms.

- We assume there are upper and lower bounds over all the variables (decision and deviation variables alike), so that we can be sure that the problem is a bounded one; this also has the advantage of accelerating the work of the simplex LP-solver.

- A *GP* problem consists of a list of goal-constraints, a list of hard constraints, a list of bounded variables, and a list of objective functions to be minimized.

  To solve a GP problem we developed the GPSolve package that implements a lexicographic (lex-min) method and several auxiliary techniques to guarantee good results when using the LP solver.
  The lex-min method is a simple inductive process that considers the next LP objective in each step, to be the next function in the objective-list (according to their given priorities) and where all prior objectives were translated into new constraints.

- For the sake of clarity, the document is written as if there is a negotiation session between two agents, Sue and *Bob*. Thus, each algorithm or procedure is implemented from the perspective of one of these agents.

- A k-level objective function is of the form $\alpha = LexMin\ or\ LexMax\{\alpha_1 \ldots \alpha_k\}$. Notice that we extended the notion of lex-min or lex-max where all levels are *minimization or maximization objectives*, so that we allow each objective level $\alpha_i \in \alpha$ to have its own type, noticing that $Max\{X\} \equiv Min\{-X\}$.

__Note__: each level is a simple expression describing a linear combination over the problem's variables. However, in the document below, we sometimes avoid describing the objective function explicitly. That is, we sometimes combine several levels into one descriptive level just for the sake of simplicity in the presentation of the model.

Note: unless otherwise stated, procedures are presented from Bob's point of view.

An example of GP Business Intentions of two Parties – An Insurance Problem

This problem deals with buying auto insurance. The parameters are: overall price, deductible, coverage, ability to choose repair shop, requirements for anti-theft devices, free towing distance, windows replacement, number of days a substitute car will be provided in case of accident and whether the customer is entitled to new replacement parts following an accident.

Min Lex problem (Bob here, a customer looking for an insurance policy)

Objective functions

G1 monetary considerations objective function, first priority level:

$$\{\pi^+ + \delta^+ + 0.01\gamma^-\}$$

G2 qualitative terms objective function, second priority level:

$$\{\rho^+ + \sigma^+ + 0.1\tau^- + \omega^+ + 0.2\nu^- + 2\mu^+\}$$

Goal Constraints

Premium constraint $P + \pi^- - \pi^+ = 1500$

Pay as less as possible Premium. Not more than 5000

Deductible constraint $D + \delta^- - \delta^+ = 1000$

Pay as less as possible deductibles. Not more than 1500

Coverage constraint $C+\gamma^{-}-\gamma^{+}=8{,}000{,}000$

Gain as much Coverage as possible. No less than 1,000,000

Repair-shop constraint $R-\rho^{+}=0$

Get free option for choosing a repair shop (R=0 – free choice)

Security constraint $S-\sigma^{+}=0$

Put as less Security equipment as possible. Not more than two anyway.

Towing constraint $T+\tau^{-}=500$

Get as much Towing distance as possible.

Windows constraint $W-\omega^{+}=0$

Get coverage for windows-repair (W=0 – covered)

Substitute-car constraint $N+\nu^{-}=364$

Get as much as possible a longer period for a substitute car.

Spare-parts constraint $M-\mu^{+}=0$

Get free choice for choosing type of spare-parts (M=0 – free-choice)

Additional bounds

$0 \le \pi^- \le 1500$

$0 \le \pi^+ \le 3500$

$0 \le \delta^- \le 1000$

$0 \le \delta^+ \le 500$

$0 \le \gamma^- \le 7,000,000$

$0 \le \gamma^+ \le 2,000,000$

Min Lex problem (Sue here, an insurance salesperson)

Objective functions (in 2 levels)

G1 monetary considerations: $\{\pi^- + \delta^- + 0.01\gamma^+\}$

G2 qualitative terms: $\{\rho^- + \sigma^- + 0.1\tau^+ + \omega^- + 0.2\nu^+ + 2\mu^-\}$

Goal Constraints

Premium constraint $P + \pi^- - \pi^+ = 2000$

Premium should not go too much under 2000. Not more than 5000

Deductible constraint $D + \delta^- - \delta^+ = 1300$

Deductibles should not exceed 1300 by too much. Not more than 1500

Coverage constraint $C+\gamma^- - \gamma^+ = 3{,}000{,}000$

Coverage should not exceed 3,000,000 by too much. No less than 1,000,000

Repair-shop constraint $R+\rho^- = 1$

Prefer not giving a free option for choosing a repair shop (R=1 – no free choice)

Security constraint $S+\sigma^- = 2$

Prefer as much Security equipment as possible.

Not more than two anyway.

Towing constraint $T-\tau^+ = 0$

Give as little Towing distance as possible.

Windows constraint $W+\omega^- = 1$

Do not provide coverage for windows-repair (W=1 – not covered)

Substitute-car constraint $N-\nu^+ = 0$

Prefer as little as possible a period for a substitute car.

Spare-parts constraint $M+\mu^- = 1$

Prefer not providing a free choice for choosing type of spare-parts (M=1 – no free-choice)

Additional bounds

$0 \leq \pi^- \leq 2000$

$0 \leq \pi^+ \leq 3000$

$0 \leq \delta^- \leq 1300$

$0 \leq \delta^+ \leq 200$

$0 \leq \gamma^- \leq 2,000,000$

$0 \leq \gamma^+ \leq 7,000,000$

GP Solver

All the basic utilities described in this document, use GP(s) as their input. Moreover, they generally construct a derived GP and solve it, to realize a desired functionality.

Generally speaking, a GP that is used during negotiation may be modified, to contain information regarding previous "achievements" of the negotiation, usually at higher levels of the objective function (see *Switch Level Utilities*).

Certain Utilities, in particular the ones dealing with negotiation support, assume that the necessary information is already contained in the GP(s) at the input.

On the other hand, the GP solver described herein makes no assumptions as to how the GP it handles was created, and solves it in a generic way. That is, it simply handles a GP model by finding, lexicographically, the minimum of its objective function levels. An important feature of solving the GP is that no pair of deviation variables which stem from the same decision variable, are both nonzero.

gp_solve

Input:

1. A *GP*: α – The problem objectives (including region-bounds)
   F_α – The goal constraints
   S_B – The hard-constraints (including the bounds)

Output:

A proposed solution P_{agent}^{new}, in which no pair of deviation variables of the same decision variable are both nonzero, which consists of:

1. A *tuple* α of the optimal objective-function values.
2. A *tuple* $T = [X \mid D]$ of the corresponding values for the deviation and decision variables, respectively.

Description:

Solves a lex-min GP via iterative calls to the Simplex algorithm.

1. For *I=1,…,k* and objective function $\alpha_i \in \alpha$ (solve a new LP)

   a. Set the objective α_i as the current LP objective.

   b. Solve the LP and find α_i^{new}, the optimal objective value for the LP problem.

   c. Add a new goal-constraint ($\alpha_i \leq \alpha_i^{new}$) into the LP.

1.1. Check for pairs of deviation variables, for the same decision variable, s.t. $(\delta_i^- > 0) \wedge (\delta_i^+ > 0)$.

If no such pairs exist proceed to step 3.1.

1.2. For every goal constraint $(f_i + \delta_i^- - \delta_i^+ = t_i)$ *s.t.* $(\delta_i^- > 0 \wedge \delta_i^+ > 0)$, add the suitable disjunction constraints to eliminate such redundancy.

1.3. Remove the constraints added in steps 1.c. above.[1]

1.4. Go back to step 1 with the same value of *k*, i.e., stay at *this* level.

3.1 Set the *Hanan-objective* as the next objective[2].

3.2 Solve the LP and find α_{Hanan}^{new} the solution of the LP with the above objective and current constraints.

4. Return.

---

[1] Usually, in ordinary GP's non-zero pair exceptions never occur. In our case, since a deviation variable may appear in more than one constraint or objective, a non-zero pair situation may happen. To overcome this difficulty, the problem is re-solved, from the beginning, with the addition of constraints on the pair of deviation variables, in order to eliminate redundant solutions with pairs of deviation variables having non-zero values

[2] *See* Hanan's Method (Pareto Efficiency).

Elimination of Non-zero Deviation Pairs

We solve GP problems in a non-direct way, by employing the *simplex* algorithm, level by level, for every level of the objectives vector.

This may easily lead to solutions where a pair of deviation variables, corresponding to the same goal will have non zero values, violating the basic requirement that $\left(\delta_i^- \cdot \delta_i^+ = 0\right)$, for every goal $\left\{g_i + \delta_i^- - \delta_i^+ = t_i\right\}$.

The solution described herein adds linear constraints to insure that at least one of the deviation-variables-pair is zero.

For every goal $\left\{g_i + \delta_i^- - \delta_i^+ = t_i\right\}$, such that $\left(\delta_i^- \cdot \delta_i^+ \neq 0\right)$, i.e., $\left(\delta_i^- > 0 \wedge \delta_i^+ > 0\right)$:

1. Add to the GP problem a new *binary* variable z_i as follows:
    a. z_i assumes only integer values
    b. Bound it: $0 \leq z_i \leq 1$
2. Add to the GP problem the following two constraints:[3]
    $\left\{\delta_i^- - ub\left(\delta_i^-\right) \cdot z_i \leq 0\right\}$ fulfilling $\left\{\delta_i^- \leq ub\left(\delta_i^-\right) \cdot z_i\right\}$
    $\left\{\delta_i^+ + ub\left(\delta_i^+\right) \cdot z_i \leq ub\left(\delta_i^+\right)\right\}$ fulfilling $\left\{\delta_i^+ \leq ub\left(\delta_i^+\right) \cdot \left(1 - z_i\right)\right\}$

    When $\left\{z_i = 0\right\}$ $\delta_i^- = 0$ is forced, when $\left\{z_i = 1\right\}$ $\delta_i^+ = 0$ is forced.

---

[3] The values $ub\left(d_i^-\right), ub\left(d_i^+\right)$ are the upper-bound constants of the $\left\{d_i^-, d_i^+\right\}$ values, respectively.

Observe that when binary variables are used we use Integer Programming techniques such as branch-and-bound on top of Simplex (as usually provided by commercial tools.

After adding all the above constraints for every pair of deviation variables, we re-solve the GP problem with the original objective function.

Hanan's Method (Pareto Efficiency)

Generating the Hanan method

This method is used to ensure that the optimal solution obtained for the GP problem, is not inferior (*see* Romero 1991). Thus, obtaining a *Pareto Efficient* solution.[4]

Finding an optimal solution for a GP problem guarantees that the list of objective functions has the minimal possible values. However, this is not always the best solution of the original problem (before it was formulated in GP terms). Specifically, it does not guarantee the optimal values of the decision variables, when there are many solutions for the same optimal objective values.

Basically, Hanan's method tries to maximize the values of the deviation-variables for which there is an implicit preference to move in their respective direction (syntactically, these are deviation variables that did not participate in any objective function).[5]

Therefore the "Hanan objective" is of the form: $\left\{-\sum\delta_i^-\right\}+\left\{-\sum\delta_j^+\right\}$ for those δ_i^-, δ_j^+ that did not appear in any objective function of the original GP problem.

Relationship between Hanan and Non-Zero Deviation Pairs

Hanan's method is one of two methods we use to enhance the *quality* of the solutions we provide for a GP problem. The other method ensures that no pair of deviations variables (participating in the same goal-constraint) will have a non-zero value for both of the variables.

---

[4] This implies, actually, that when the user defines in the GUI an *indifference range*, we assume that the user will be happier if we improve the solution within this range. For interval goal programming, some of the deviation variables need to appear with zero coefficients in the objectives.

[5] We may consider the lex-min objective-vector as a requirement to minimize the damage of deviating from the targets set on the goal-constraints in the *undesirable* direction (minus or plus). Thus, Hanan tries to maximize the benefit of deviating from the target in the *desirable* direction.

Next, we show that Hanan's method cannot eliminate, nor prevent, deviation variables from fulfilling the requirement: $\delta_i^+ \cdot \delta_i^- = 0$ (cases of non-fulfillment do exist). This explains why we need to handle such cases explicitly.

Example 1:

This example shows that Hanan's method cannot eliminate a solution, where a pair of deviation variables has no- zero values.

$$
\begin{aligned}
&Min\_Lex\{-\delta_i^-\}\\
&X + \delta_i^- - \delta_i^+ = 100\\
&X = 70\\
&\delta_i^- \leq 150\\
&\delta_i^+ \leq 150\\
&Hanan : Min\{-\delta_i'\}
\end{aligned}
$$

- Solving the lex-min above, we get the solution:

$\{X = 70\}, \{\delta_i^- = 150\}, \{\delta_i^+ = 120\}$

- Before solving Hanan we add the constraint, which corresponds to the level we just solved: $\{-\delta_i^- \leq -150\}$. This is in preparation for

Example 2:

Hanan's method forces a solution where both deviation variables are non-zero.

$$
\begin{aligned}
&MinLex\{-\delta_i^-\}\\
&X + \delta_i^- - \delta_i^+ = 100\\
&50 \leq X \leq 150\\
&\delta_i^- \leq 50\\
&\delta_i^+ \leq 50\\
&Hanan : Min\{-\delta_i^+\}
\end{aligned}
$$

- Solving the lex-min above, we get the solution:

$\{\delta_i^- = 50\}$. And possibly: $\{X = 50\}, \{\delta_i^+ = 0\}$.

- Before solving Hanan we add the constraint:

$\{-\delta_i^- \leq -50\}$ in preparing for the next level.

General Purpose Utilities

Suggest

Input:

1. A *GP*: α – The problem objectives
   F_α – The goal constraints
   S_B – The hard-constraints
2. *Mode* of operation: [Optimal, Worst, Any, Percent, Average]
3. [An optional percentage ρ, for the Percent mode]
4. [An optional reference value α^0 to improve, for the Percent mode]

Output:

A proposed solution P_B^{new}, consisting of

1. A *tuple* $T = [D \mid X]$ of values for the deviation, *D*, and decision variables, *X*.
2. A *tuple* α of the objective-function values.

Description:

The function suggests a solution, according to the mode of operation, as follows:

Optimal

Derives the optimal solution for the GP. [6]

---

[6] The utility layer has no preference to minimization problems over maximization ones.

1. Call *gp_solve(GP)* and return its result[7]: $T^{*}=\left[D^{*}\mid X^{*}\right]$ and α^{*}.

Worst

Derives the maximum solution for the GP (recall that all GP variables are bound).

1. Invert the objective functions in the GP formulation. That is, at every level change the *Min* operator to *Max* (and vice versa, depending on its original direction).
2. Call *gp_solve(GP)* and return its output: $T_{*}=\left[D_{*}\mid X_{*}\right]$ and α_{*}.

Any

Derives an arbitrary feasible solution.

1. Prepare a single objective function as follows.

   For every pair of deviation variables of the original GP:

   Randomly choose one of them, and randomly assign a coefficient for the chosen deviation variable in the range [0,b] where b is a system parameter. Insert the deviation variable multiplied by its coefficient into the objective function. The coefficient for the other member of the pair is set to zero.

2. Call *gp_solve(GP)* and return its output.[8]

Percent (ρ) [9]

Derives a solution that is worse than α^{0} values by no more than ρ %. The current solution reduces each objective at a time, using the same percentage ρ %

---

[7] We use * in superscripts to indicate best and in subscript to indicate worst.

[8] The function must generate a feasible solution, unless the goal-constraints and bounds have no feasible solution.

[9] This is considered as a general and simple Improve function. It also works Level-by-Level.

for all objectives. Other variations are possible (e.g., do it only for the first, and most important, level).

1. Add to the *GP* the following goal-constraints and bounds:
    a. For every objective function α_i and its value α_i^0 [10]
        - Add the hard constraint: $\alpha_i + \delta_i^- - \delta_i^+ = \alpha_i^0 + \rho \cdot |\alpha_i^0|$
2. Replace the objective-function vector of *GP*:
    a. Insert the function $\sum_{i=1}^{k}(\omega_i^+\delta_i^+ + \omega_i^-\delta_i^-)$ as the first priority level; here the weights ω_i^+, ω_i are such that the lower *i* is, the higher the weight, e.g., use powers of ten.
    b. Set the rest of the priority levels according to the original α objective function (that is, each function in the original objective is pushed one level down, in order).
3. Call *gp_solve(GP)* and return its output.

Average

Derives a solution that is as close as possible to the average (between the optimal and the worst solutions) of the objective function values[11].

1. Find the optimal solution X^*, α^* and the worst solution X_*, α_*
2. Add to the *GP* the following goal-constraints and bounds:
    a. For every objective function α_i: [12]

---

[10] The α_{Bi}^{last} values are taken from P_B^{last}.

[11] The derived solution will be feasible since it is generated by a GP formulation where feasibility is guaranteed.

- Add the goal constraint: $\alpha_i + \delta_i^- - \delta_i^+ = \left(\alpha_i^* + \alpha_{i*}\right)/2$

3. Replace the objective-function vector of *GP*:

    a. Insert the function: $\sum_{i=1}^{k}\left(\omega_i^+\delta_i^+ + \omega_i^-\delta_i^-\right)$ as the first priority level; here the weights ω_i^+, ω_i^- are such that the lower *i* is, the higher the weight, e.g., use powers of ten.

    b. Set the rest of the priority levels according to the original α objective function (that is, each function in the original objective is pushed one level down, in order).

4. Call *gp_solve(GP)* and return its output.

Rank

Input:

1. A *GP*: α – The problem objectives
    F_α – The goal constraints
    S_B – The hard-constraints
2. A set of *h* tuples of values $\left\{T^j = D^j \mid X^j\right\}$[13] 1≤j≤*h*

Output:

1. A sorted set of *h* tuples of values: $\left\{T^{j'} = D^{j'} \mid X^{j'}\right\}$ each with its rank number.

---

[12] The $\alpha_{B(i)}^{last}$ values are taken from P_B^{last}.

[13] The tuples may be partial or complete.

Description:

Calculates the objective function values for each tuple, and performs a lexicographical ordering of the objective vectors to sort the tuples. As a result, the tuples are ordered from most preferred to least preferred.

1. *For every input tuple* T^j, $1 \leq j \leq h$
    a. *If the tuple* T^j *is partial, complete it to the optimal possible values.* [14]
    b. *Calculate the objective function values* (α^j) *– each element in the* α *vector is a weighted sum of deviation values.*
    c. *If the tuple is not feasible, then discard it from the sorting process.*
2. *Lexicographically, sort the tuples* $\alpha^1 \ldots \alpha^h$ *into* $\alpha^{1'} \ldots \alpha^{h'}$
3. *Re-arrange and return the tuples* $T^{1'} \ldots T^{h'}$ *according to the arrangement of* $\alpha^{1'} \ldots \alpha^{h'}$ *(k'<k is possible in case some input tuples are discarded as not feasible)*

Choose

Input:

1. A *GP*: α – The problem objectives
   F_α – The goal constraints
   S_B – The hard-constraints
2. A set of *h* tuples of values $\{T^j = D^j \mid X^j\}$[15] $1 \leq j \leq h$
3. The number *m* of best tuples to be chosen (out of *h*).
4. A maximum threshold vector *H* on the objective values of the tuples.[16]

---

[14] This is done by calling *Complete(GP,* T^j, *Optimal)*

Output:

1. A sorted set of m tuples of values $\{T_i' = [D_i' \mid X_i']\}$ of values, each with its rank number.

Description:

The purpose of this function is to choose the best m good-tuples out of the suggested input tuples T. A tuple is considered as "good", when its objective-vector value does not exceed the maximal threshold H.

1. *For every input tuple* T^j, $1 \leq j \leq h$
   a. *If the tuple* T^j *is partial, complete it to the optimal possible values.* [17]
   b. *Calculate the objective function* α^j *values.*
   c. *If the tuple is not feasible, then discard it from the sorting process.*
   d. *If* $\alpha^j < H$[18], *then discard it from the sorting process.*
2. *Lexicographically* sort[19] *the tuples* $\alpha^1 \ldots \alpha^h$ *into* $\alpha^{1'} \ldots \alpha^{h'}$
3. *Re-arrange and return the best up to* m *tuples* $T^{1'} \ldots T^{m'}$ *according to the arrangement of* $\alpha^{1'} \ldots \alpha^{h'}$

Max_Rank

As mentioned in the previous section, the Choose function should *sort* the objective functions, to find the best *m* tuples.

---

[15] The tuples may be partial or complete.

[16] This threshold indicates whether a tuple is too bad to be considered by the Choose function.

[17] This is done by calling *Complete(GP,* T^j, *Optimal)*

[18] Lexicographical comparison.

[19] This may require sorting or just finding the best tuples, see the next section about Max_Rank

However, suppose m=1, in this case we are required to return the best tuple. Thus, it is enough to solve the minimum (or maximum) problem, rather than the sort problem.
We chose an arbitrary threshold of seven, so that if $m \leq 7$, we solve a max/min problem, and otherwise we sort the tuples.

Therefore, this function is needed only for efficiency reasons. It is considered as an auxiliary function.

The algorithm is a trivial extension that finds the maximal value within a set of values:

Sketch of the Max_rank algorithm:

Let the sorted set, *Max7*, hold the current best up to seven tuples out of those examined thus far. (The number 7 could be replaced with any other number that is judged as "small" in the problem domain.)

```
1. For every tuple $\alpha^i, i \in [1..h]$
    a. If $\alpha^i$ is better than one of the vector in Max7, then:
        i. Delete the lowest, i.e. worst, member of the set.
        ii. Add $\alpha_i$ to Max7.
```

Complete (handling of partial tuples)

Input:

1. A *GP*: α – The problem objectives
   F_α – The goal constraints
   S_B – The hard-constraints

2. A partial tuple of values for decision variables [20]. Each value is associated with a {generous, tough} flag where "generous" means that the program is allowed to change the respective value to overcome potential infeasibilities and "tough" means that no change is allowed.
3. *Mode* of operation: [Optimal, Worst, Any, Percent, Average]
4. [An optional percentage ρ, for the Percent mode]
5. [An optional reference value α^0 to improve, for the Percent mode]

Output:

A proposed solution P_B^{new}, which consists of

1. A complete tuple $T = [D \mid X]$ of values for the GP problem variables
2. A *tuple* α of the objective-function values

Description:

This function provides a solution that keeps the given values for those decision variables that were specified as inputs, and assigns values to the rest of the variables according to the mode of the operation.[21]

1. *For every decision variable x_i with given value X_i (input values X_i will not change)*
   a. *Add to the GP a new hard-constraint:* $(x_i = X_i)$[22].

---

[20] The fixed variables are provided explicitly, that is, for every fixed variable, the function is provided with the variable name and its value. Hence, there is no need to indicate which variables are left undetermined.

[21] For example, if the mode of operation is *Optimal*, the function attempts to find the minimal values possible for the objective function, while keeping the values for the partial input tuple unaltered. In a similar way, the function completes the partial tuple towards a *Worst*, *Any*, *Average*, and *Percent* solution.

[22] These are *hard-constraints* with no deviation values allowed; therefore, the solver is forced to assign to the input variables the given values.

2. Call *gp_solve(GP, mode)* and return an output if possible (i.e., the supplied fixed values do not lead to a contradiction). Otherwise, an exception message is provided together with a solution as described in the next step.

3. Turn each fixed value X_i for variable x_i into a goal constraint of the form $x_i + \delta_i^- - \delta_i^+ = X_i$ and add a new objective of the form $\sum_{i=1}^{k}\left(\omega_i^+\delta_i^+ + \omega_i^-\delta_i^-\right)$ as the first priority level; here we assume that the variables x_i are ordered in their order of appearance in the objective functions of α, the weights ω_i^+, ω_i^- are such that the lower i is, the higher the weight, e.g., use powers of ten (if such an order is not provided, the weight is 1 for all, this is essentially the same concept used in prescribing "stay close" in the utilities supporting the ignorant mode) . The weights associated with variables classified as "tough", are multiplied by a system constant $s>1$.

4. Set the rest of the priority levels according to the original α objective *function* (that is, each function in the original objective is pushed one level down, in order).

Negotiation Support Utilities

Switch Level Utilities

Switch Level

Level-by-Level negotiation strategy: Motivation

1. *Negotiation may take many shapes and forms, over the whole problem at once, or some specific subset of the decision variables, or specific objective-levels. Here we detail how the operations of the various Improve functions are altered by specifying bounds for certain objective levels, ignoring certain levels and minimizing (or maximizing) other objective levels.*

2. *Crossover effect*

This problem has different effects in each of the improve algorithms.

As the negotiation proceeds over more than one level, it may proceed faster in one level than the other. Furthermore, the agents will not terminate in agreement until they agree upon the values of the first level, the most important one.

Therefore, if the negotiation proceeds faster over the lower levels, an agent, say Sue, may start suggesting values for the lower values, such that Bob already agree upon, or worse, Bob already suggested better values from Sue's viewpoint.

Implementation Required

The negotiation takes place upon one level at a time, the same level for both agents. That is, the agents negotiate only over the first level of both agents, then after agreement on the values of their first level objective values, they should proceed to negotiate over the second level, etc.

The agreement has the *selfish* meaning, of how far is the opponent willing to give up in terms of my objective level-value. E.g. suppose that two agents agree on the i^{th} level with objective values α_i, β_i for Bob's and Sue's values respectively, then Sue relies on the fact that Bob is willing to agree for Sue's value β_i in any final agreement they have.

Switch Level {Ignorant, Any}

If the agent, say Bob, is ignorant, then after the agreement over the i^{th} level objective value, Bob will add to his GP formulation, the hard constraint as follows:

$$\alpha_{Bi} \leq \alpha_i .$$

Note: the above implies that when the negotiators are ignorant, they may produce after switching a level infeasible offers for their opponents.

Switch Level {Informed, Any}

If the agent, say Sue, is knowledgeable, then the agent adds constraints on both values as follows:

$\beta_{Si} \leq \beta_i$;

$\alpha_{Si} \leq \alpha_i$

First-Offer {Informed, any} (Sue's viewpoint)

Input:

1. Agent's *GP:* (GP_{β})

   β – Sue's objectives

   F_{β} – Sue's goal constraints

   S_S – Sue's hard-constraints

2. The opponent's GP: (GP_{α})

   α – Bob's objectives

   F_{α} – Bob's goal constraints

   S_B – Bob's hard-constraints

3. The new level *k* of negotiation.

Output:

A proposed solution P_S^{new}, which consists of:

1. A *tuple* $T' = [D' \mid X']$ of values for the variables.
2. A *tuple* β' of the corresponding objective-functions values.

Description:

This utility is used by the 1-1 negotiation mechanism to construct first offers for negotiation over a new level.

Basically, the function combines the GPs of both agents, and tries to find the best offer for the agent then the worst offer for the opponent. This is done level-by-level, working on the first level of the agent objective, followed by the first level of the opponent's objective, then the second level of both objectives, etc.

Notice that when the function is called after agreeing on the k^{th} *level, then both GPs already contain the achievement values of the previous k objective levels.*

1. Add to (GP_β) the following goal-constraints and bounds:

   1.1. Add the opponent's goal-constraints and bounds (GP_α)

2. Replace the objective-functions vector of (GP_β):

   1.2. The k^{th} levels of both agents will be formulated in the first-offer objective function as follows:

$$\begin{array}{lll} Min\_lex & (2k) & \beta_{St} \\ Max\_lex & (2k+1) & \alpha_{S_t} \\ & \cdots & \end{array}$$

   1.3. Set a suitable Hanan objective level as the last objective function.

   Note: If one of the agents has fewer levels than the other, then the extra levels will still be added at the end of the objective function.

3. Call *gp_solve*(GP_β) and return its output.

First-Offer {Ignorant, any} (Bob's viewpoint)

Input:

1. Agent's *GP*: (GP_α)
   - α – Bob's objectives
   - F_α – Bob's goal constraints
   - S_B – Bob's hard-constraints
2. A tuple X^{last} of values for the decision variables that caused the agreement on the last level.
3. The new level k of negotiation.

Output:

A proposed solution P_S^{new}, which consists of:

1. A *tuple* $T' = [D' \mid X']$ of values for the variables.
2. A *tuple* α' of the corresponding objective-functions values.

Description:

This utility is used by the 1-1 negotiation mechanism to construct first offers for negotiation over a new level.

The Ignorant agent computes the very first offer of the whole negotiation by calling the Suggest(Optimal) function. Moreover, notice that the knowledgeable agent uses the First-Offer function for the very first offer, and for the first offers after switching (upon agreement) a level. The Ignorant agent showed that he requires special care on this matter.

1. Add to (GP_α) the following goal-constraints and bounds:
   a. For every decision variable x_i and its input value $X_i \in X^{last}$
      - Add the goal constraint: $x_i + \gamma_i^- - \gamma_i^+ = X_i$

2. Replace the objective-function vector of (GP_α):
    a. Set the first priority ordering for functions according to the original α objective function
    b. Set the last priority function as $\sum_{i=1}^{k} V_i^+ \gamma_i^+ + V_i^- \cdot \gamma_i^-$. *(Stay close)*.
    c. Set a suitable Hanan objective level.
3. Call *gp_solve*(GP_α) and return its output.

Motivation

The difference between this function and the Suggest (Optimal) one is in the stay-close constraints and objective-level. This is required to keep the Ignorant tuned with the *correct direction* of the decision variables values towards the agreement tuple X^{last}. Otherwise, both Ignorant agents negotiation may get into many rounds of infeasible values after every level agreement, because each Ignorant agent will keep his value and totally *forget* the correct deviation required for a decision variable, to get the agreement of the other agent.

Level-by-level initialization

When switching from one level to the next there is a need to update some of the system parameters in light of what was agreed upon in the previous level(s). Specifically, the My_best and My_worst values for the present lexicographical level (denoted as $\square^*$ and $\square_*$) might change as a result of the $\square^*$ values that were achieved at upstream levels and the presence of hard constraints that link variables across more than one level. The following example will illustrate the need for this update.

Example:

$$
\begin{array}{ll}
(L_1) & Min \quad \delta_x^- + 2\cdot\delta_x^+ \\
(L_2) & Min \quad \delta_y^- + \delta_y^+ + 3\cdot\delta_z^- + 2\cdot\delta_z^+ \\
 & s.t. \\
 & x + \delta_x^- - \delta_x^+ = 50 \\
 & y + \delta_y^- - \delta_y^+ = 80 \\
 & z + \delta_z^- - \delta_z^+ = 70 \\
 & x + y + z = 200 \\
 & x, y, z, \delta_x^-, \delta_x^+, \delta_y^-, \delta_y^+, \delta_z^-, \delta_z^+ \geq 0
\end{array}
$$

Assume that the opposing party was also interested only in the x variable at L_1 and that his target was 20. Let 's suppose that the outcome of the negotiations in L_1 was $\alpha_1^* = 20$ (as a result of the values x=30, $\delta_x^- = 20, \delta_x^+ = 0$). We incorporate the constraint $\alpha_1 = 20$ into the goal constraint of x in L_2. Notice that this is not equivalent to setting x=30 since the L_2 program has an option of choosing between x=30 and x=60 as both of these values satisfy the additional constraint on α_1. Regardless of which of these two values is selected, the hard constraint will no longer fit the targets of variables y and z. Thus, while the original $\square^*$ value for L_2 was zero, the updated value is $\alpha_2^* = 10$ (resulting from the values x=60, y=z=70, $\delta_y^- = 10, \delta_y^+ = \delta_z^- = \delta_z^+ = 0$).

Notice that to find the updated values of $\square^*$ and $\square_*$ for an ignorant agent we do not add an equality-type constraint rather than an inequality-type constraint (i.e., $\alpha_1 = 20$). Otherwise, if we were to add the constraint $\alpha_1 \leq 20$, there would have been no effective changes in $\square^*$ and $\square_*$ of our current level (since $\alpha_1 = 0$ would have still been feasible). This difficulty is not present when the agent is aggressive since the latter incorporates both the constraint representing his own α_1 value and the β_1 value of his opponent in his GP. Thus, in aggressive-improve implementation, both of these additional constraints can be written as inequalities.

Improve Utilities

In these utilities, we differentiate between versions specialized to parties' state of knowledge. Here, ignorant means lacks knowledge of the other parties GP. Informed means a party that is knowledgeable concerning the GP of the other party. The layer that calls these improve routines decides as to which improvement style is preferable. The layering is part of the improve utility feature but is not discussed further herein. Intuitively, an ignorant party may find it difficult to improve on his previous offer for the other party due to his ignorance as to the other party's GP. The opposite holds for an informed party. However, an informed party will usually not blindly improve the offer for the other party, usually some constraints on the worsening of his position, as judged by his very own GP, will be applied.

We note that these improve procedures may be turned into "worsening" procedures by "changing directions", for brevity we do not detail these changes.

Improve {Ignorant, any} (Bob's viewpoint)

Input:

1. A *GP*: α – The problem objectives
   F_α – The goal constraints
   S_B – The hard-constraints
2. P_B^{last} -- The reference proposal, that is the previous offer Bob made which serves as a reference point.
3. A tuple X_S^{last} of values for the decision variables that appeared in Sue's last offer.
4. A percentage ρ.
5. α^* – The optimal objective values of Bob.
6. α_* – The worst values of Bob.
7. *gap_flag* – *A* flag to determine the way to calculate the new objective function values. Possible values are: *fixed, dynamic*.

8. *Max_gap* – The maximal value allowed for changing the objective function.
9. *Min_gap* – The minimal value allowed for changing the objective function.
10. *gc* – The gap constant for the fixed *gap_flag*.

Output:

A proposed solution P_B^{new}, which consists of

1. A *tuple* $T' = [D' \mid X']$ of values for the variables.
2. A *tuple* α' of the corresponding objective-functions values.

Description:

Given input values (corresponding to objective function α), this utility finds a new objective function values α', that worsens the values of P_B^{last} *by at most* $\rho\%$ [23]*. The worsening is done with respect to the differential value* $(\alpha_S^{last} - \alpha^*)$ *where the first value is the value of the counter-proposal X, and the second is the optimal value.*

1. If the tuple X_S^{last} is not complete, call *Complete(GP,* X_S^{last}*, Optimal)*[24]
2. Add to (GP) the following goal-constraints and bounds:
   a. Calculate the objective-values $\alpha_S^{last} = (\alpha_{S1}^{last}, ..., \alpha_{Sk}^{last})$[25] where k is the number of levels in the lexicographical objective function.

---

[23] The same percentage may be used for all levels of the objectives list.

[24] This is required so that we can evaluate α_S^{last} (see next footnote).

[25] Calculated according to the completed tuple X_S^{last}.

b. For the current objective function-level α_i[26] and its value α_{Bi}^{last},[27] Add a hard constraint to set the new range of the objectives as follows:

 i. Calculate the *gap value* of the increase in the region value, this is done according to the value of *gap_flag*;
 currently we support two modes of operation:

Fixed: $gap = \left(\alpha_{*i} - \alpha_i^*\right)/gc$; here *gc (gap constant), e.g., 25.*

Dynamic: $gap = \left(\alpha_{Si}^{last} - \alpha_{Bi}^{last}\right)$; differential between proposals.

 ii. Cut off the value of *gap*, if necessary:

If $gap > Max\_gap$ then set $gap = Max\_gap$.

If $gap < Min\_gap$ then set $gap = Min\_gap$.

 iii. Add the hard constraint: $\alpha_i + \delta_B^- - \delta_B^+ = \alpha_{Bi}^{last} + \rho \cdot gap$.

c. For every decision variable x_j and its input value $X_j \in X_S^{last}$[28]

- Add the goal constraint: $x_j + \gamma_j^- - \gamma_j^+ = X_j$

3. Replace the objective-function vector of (GP_α):

a. Set the first priority level function as $\{100 \cdot \delta_B^+ + \delta_B^-\}$. This objective tries to achieve the improvement value for the opponent. If the new value cannot be reached, then try to improve further in order to avoid not improving at all.

b. Set the second priority function as $\sum_{j=1}^{m}\left(V_j^+\gamma_j^+ + V_j^- \cdot \gamma_j^-\right)$ where m is the number of decision variables (the length of the vector x). This objective is called *stay close*. The idea is to stay as close as possible, under the constraints, to Sue's last offer. The weights V_j^+, V_j^- are set as follows. The last two offers of Sue are compared. For each decision variable, the percentage increase or decrease from the previous offer is calculated. The weights are set between c_1 and c_2 (typical values may be 1.0 and 2.0). For a minor change of say up to 1%, the weight is set as c_1. For a major change, say above 75%, the weight is set as c_2. In the intermediate range, the weight for a percentage change *p* is set proportionately between c_1 and c_2, that is $c_1+(p/(75-1))*(c_2-c_1)$.

---

[26] See Level-by-Level negotiation strategy.

[27] The $\alpha_{B(i)}^{last}, \alpha_i^*$, and α_{*i} values are taken from P_B^{last}, P_H^*, and P_{*B} respectively.

[28] This step is concerned with the original X_S^{last} (before completing it), as there is no reason why we should constrain the algorithm to stay close to values generated by the complete function (i.e. those values that were not provided by the opposite agent).

c. Set a suitable Hanan objective level as the last objective function.

4. Call $gp\_solve(GP_\alpha)$ and return its output.

The complete formulation of the model is therefore:

(1) $$\text{Lex\_min } \{100\cdot\delta_B^+ + \delta_B^-\}, \left\{\sum_{j=1}^{m}\left(V_j^+\cdot\gamma_j^+ + V_j^-\cdot\gamma_j^-\right)\right\}, \alpha_{i+1}, \alpha_{i+2}, \ldots, \alpha_k$$

Subject to:

(2a) Ignorant 1: $\alpha_B^{new} + \delta_B^- - \delta_B^+ = \alpha_B^{last} + \rho_B\cdot(Ign1\_gap)$ *Worsen mine: Dynamic gap*

(2b) Ignorant 3: $\alpha_B^{new} + \delta_B^- - \delta_B^+ = \alpha_B^{last} + \rho_B\cdot(Ign3\_gap)$ *Worsen mine: Static gap*

(3) $x_{Bj}^{new} + \gamma_j^- - \gamma_j^+ = x_{Bj}^{last} + \rho_B\cdot(Max\_gap)$ *Stay close.*

$$Max\_gap = \begin{cases} (x_{Sj}^{last} - x_{Bj}^{last}) & \text{if } \left|x_{Sj}^{last} - x_{Bj}^{last}\right| \geq \left|x_{sj}^{last-1} - x_{sj}^{last}\right| \\ (x_{sj}^{last-1} - x_{sj}^{last}) & \text{if } \left|x_{Sj}^{last} - x_{Bj}^{last}\right| < \left|x_{sj}^{last-1} - x_{sj}^{last}\right| \end{cases}$$

(4) $\alpha_B^{new} \leq R_B$ *Protect thyself*

(5) $x_B^{new} \in F_\alpha \qquad x_B^{new} \in S_B$

(6)

$$Ign1\_gap = \begin{cases} MinGap = \frac{(\alpha_* - \alpha^*)}{20}, if\ \alpha_S^{last} - \alpha_B^{last} \le MinGap \\ MaxGap = \frac{(\alpha_* - \alpha^*)}{5}, if\ \alpha_S^{last} - \alpha_B^{last} \ge MaxGap \\ \alpha_S^{last} - \alpha_B^{last}, elsewhere \end{cases}$$

$$Ign3\_gap = \frac{(\alpha_* - \alpha^*)}{10}$$

Further explanation

(1) The first level of this objective is related to the "worsen-mine" constraint (either (2a) or (2b)). The penalty for under-deviation is much larger than that of over-deviations since in general we intend to worsen our situation. The need to incorporate δ_B^- is to enable the program to escape from infeasible situations (e.g., when due to integer variable requirements it is impossible to worsen α but possible to slightly improve it. The second level handles deviations from the stay-close constraint. Then, from level 3 onwards we minimize (lexicographically) the remaining alpha values from i+1 through k. The minimization of these additional alpha values is optional.

(2) An "Ignorant" agent can't improve his opponent's situation since he is not aware of the latter GP. Thus, by worsening his own status (albeit in a moderate manner) he implicitly assumes that his opponent will gain something. There are two versions for this worsening procedure that are further explained through the definitions of the gaps in (6).

(3) Here, since we lack other information, we try to gain whatever we can from the recent moves of the opponent. Observing his last two offers we see the "gradient"

of change in the values of x_j. If he makes a small move on that variable we may conclude that this variable is important to him and therefore we are reluctant to make a relatively large step towards him (as might happen when the □ factor is multiplied by a difference that might be large).

Notice that if the opponent performs a big step in the value of some variable, X_j, then the value $x_{Bj}^{last} + \rho(x_{sj}^{last-1} - x_{sj}^{last})$ may be even better the value x_{sj}^{last} suggested by the opponent. However, the stay close is expressed in the second level of the objective, and is subject to the opponent's objective value allowed which is expressed in the first level of the objective function.

(4) Protection against "over-enthusiasm" in the worsening action of (2).

(5) The ordinary GP constraints.

(6) The gap we implement in (2) can either be static or dynamic. The dynamic option (lgn_gap1) is based on the current difference between the last offer made by the other party and my own best solution. A certain proportion (□) will be taken of that difference unless it becomes too small (at which point we truncate it with the *MinGap* parameter) or too large (at which point we truncate it with the *MaxGap* parameter). The static option uses a fixed value (a certain proportion of the difference between best and worst solutions) to determine the step size taken in (2).

Improve-Aggressive {Informed, any} (Sue's viewpoint)

Input:

1. Agent's *GP*: (GP_β)

   β – Sue's objectives
   F_β – Sue's goal constraints
   S_S – Sue's hard-constraints

2. P_S^{last} – The reference proposal, that is the previous offer Sue made which serves as a reference point.
3. The opponent's GP:[29] (GP_α)

   α – Bob's objectives
   F_α – Bob's goal constraints
   S_B – Bob's hard-constraints

4. The opponent agent's optimal result α^* [30].
5. A tuple X_β^{last} of values for the decision variables
6. A percentage ρ_{BS} for the maximum improving to the opponent objective.
7. A percentage ρ_{SS} for the maximum worsening of the agent objective.
8. Agent's optimal result β^*.

Output:

A proposed solution P_S^{new}, which consists of:

2. A *tuple* $T' = [D' \mid X']$ of values for the variables.

---

[29] The opponent's information is subject to the mechanism's strategy. For example, the opponent may not provide all of his preferences (constraints or objectives), or even may not provide true preferences!

[30] Calculated according to our knowledge about the opponent (see previous footnote).

3. A *tuple* β' of the corresponding objective-functions values.

Description:

Given input values (corresponding to Sue's objective function β*), this utility constructs a new objective function* β' *that improves the values of the objective function value for Bob by at least* $\rho\%$[31]*. Taking into consideration Bob's constraints and goals as represented by his GP, we try to improve the value of the opponent's objectives in a controlled manner.*

---

[31] The same percentage may be used for all levels of the objectives list.

2. Add to (GP_β) the following goal-constraints and bounds:

   a. Calculate the objective-values $\alpha_S^{last} = \left(\alpha_{S1}^{last},\ldots,\alpha_{Sk}^{last}\right)$[32]

   b. For the current objective function-level α_i and its value α_{Si}^{last}

      - Add the goal constraint:[33] $\alpha_i + \delta_i^- - \delta_i^+ = \alpha_{Si}^{last} - \rho_{BS} \cdot \left(\alpha_{Si}^{last} - \alpha_i^*\right)$[34]

   c. For every decision variable x_j
      - Add the goal constraint:
        $x_j + \gamma_j^- - \gamma_j^+ = X_{Sj}^{last} - \rho_{BS} \cdot \left(X_{Sj}^{last} - X_{Bj}^{last}\right)$

   d. - Add the goals F_α and bounds S_B of the opponent agent's (GP_α).[35]

   e. For the current objective function-level β_i and its value β_{Si}^{last}

      - Add the goal constraint: $\beta_i \le \beta_{Si}^{last} + \rho_{SS} \cdot \left(\beta_{*i} - \beta_{Si}^{last}\right)$

3. Replace the objective-functions vector of (GP_β):

   a. Set the first priority function as $\left(a \cdot \delta_i^+ + b \cdot \delta_i^-\right)$ where a and b are weights that reflect Sue's preferences for over or under improvements for Bob (default values are a=b=1).

   b. Set the following priority functions according to the original β objective functions from level i onwards.

---

[32] Calculated according to the last tuple proposed by the Sue X_S^{last}.

[33] The sign of the addition $|\alpha_i| \cdot p\%$ depends on the flag, that is, a negative sign for improve, and positive sign for reduce.

[34] See Level-by-Level negotiation strategy.

[35] Both agents must agree on the decision variables representation.

c. Set the next priority function as $\sum_{j=1}^{m}\left(V_j^{+}\gamma_j^{+}+V_j^{-}\cdot\gamma_j^{-}\right)$, the weights V_j^{+}, V_j^{-} are set as in the previous Improve routine.

d. Set a suitable Hanan objective as the last level

4. Call *gp_solve*$\left(GP_\beta\right)$ and return its output.

The complete formulation of the model is therefore:

(1) Lex_min $\left\{\delta_i^{+}+\delta_i^{-}\right\},\left\{\beta_i,\beta_{i+1},\ldots,\beta_k\right\},\left\{\sum_{j=1}^{m}\left(V_j^{+}\cdot\gamma_j^{+}+V_j^{-}\cdot\gamma_j^{-}\right)\right\}$

Subject to:

(2) $\alpha_S^{new}+\delta_i^{-}-\delta_i^{+}=\alpha_S^{last}-\rho_S\cdot\left(\alpha_S^{last}-\alpha^{*}\right)$ *Improve other*

(3) $x_{Sj}^{new}+\gamma_j^{-}-\gamma_j^{+}=x_{Sj}^{last}-\rho_S\cdot(x_{Sj}^{last}-x_{Bj}^{last})$ *Stay close*

(4) $\beta_S^{new}\leq R_S$, Explicitly: $\beta_i\leq\beta_{Si}^{last}+\rho_{SS}\cdot\left(\beta_{i*}-\beta_{Si}^{last}\right)$ *Protect thyself*

(5) $\alpha\in F_\alpha,\beta\in F_\beta,L_j\leq X_j\leq U_j$ *Original GP constraints*

Further explanations:

(1) The first level of the objective tries to improve the current objective level of the opponent party (here, Bob's □). A user-given proportion dictates the improvement. Sometimes it won't be possible to improve (e.g., due to discontinuities) and then we allow worsening of the opponent's objective – but, the penalty on moving in that direction is much larger than the deviation under the new target for his alpha value.
The second level tries to improve my own objective (here, Sue's □).
The third level prevents decision variables that are not strongly affected by the first two levels (say, because they are associated with less important levels of the GP) from either "jumping around" in a senseless manner or from approaching too fast the values desired by the other side.

(2) The assignment of the new values for the decision variables that Sue is going to offer, x_{Sj}^{new}, will yield for Bob a new value of α that will be better (smaller) than the previous value offered by Sue. The improvement for Bob is determined by a proportion ρ_S of the distance between the last Sue's offer and the best value possible from Bob's viewpoint (α^*).

(3) This constraint relates the new offer to the last offer by the same party with some adjustments to accommodate for the preferred values of the other party. Thus, for example, if the value for x_j in Sue's last offer was higher (lower) than the corresponding value in Bob's last offer, the new value offered by Sue will be smaller (larger) than that offered earlier.

(4) This constraint protects Sue from situations in which the direction determined by the first constraint (that improves the outcome for the opponent) hurts his own objectives too severely. Notice that the protection value is calculated is a similar way to the improve-other value, according to the interval left for progress. I.e. the

improvement is calculated as a percent according to the interval $\left(\alpha_S^{last} - \alpha^*\right)$ while the worsening (protection) value is calculated according to the interval $\left(\beta_* - \beta_S^{last}\right)$

(5) The rest of the ordinary constraints: α, β defined through F_α, F_β, respectively and the upper and lower bounds on the decision variables.

Improve-Cooperative {Informed, any} (Sue's viewpoint)

Input:

1. Agent's *GP*: (GP_β)

   β – Sue's objectives

   F_β – Sue's goal constraints

   S_S – Sue's hard-constraints

2. P_S^{last} – The reference proposal, that is the previous offer Sue made which serves as a reference point.

3. The opponent's GP:[36] (GP_α)

   α – Bob's objectives

   F_α – Bob's goal constraints

   S_B – Bob's hard-constraints

4. The agent's optimal result β^* [37].

5. A tuple X_β^{last} of values for the decision variables.

6. A percentage ρ.

Output:

A proposed solution P_S^{new}, which consists of

1. A *tuple* $T' = [D' \mid X']$ of values for the variables.
2. A *tuple* β' of the corresponding objective-functions values.

---

[36] The opponent's information is subject to the mechanism's strategy. For example, the opponent may not provide all of his preferences (constraints or objectives), or even may not provide true preferences!

[37] Calculated according to our knowledge about the opponent (see previous footnote).

Description:

Given input values (corresponding to objective function β), this utility constructs a new objective function β' that worsens the values of Sue's objective function by at most ρ %[38]*. Taking into consideration the opponent's constraints and goals represented by the opponent's GP we try to improve the value of the opponent's objectives the best we can, as long as we do not exceed an upper limit on our own objective values.*

---

[38] The same percentage may be used for all levels of the objectives list.

1. Add to (GP_β) the following goal-constraints and bounds:[39]
   a. For every objective function β_i and its value β_{Si}^{last} [40]
      - Add the hard constraint: $\beta_{Si} \leq \beta_{Si}^{last} + \rho \cdot \left(\beta_{Si}^{last} - \beta_i^*\right)$ [41]
   b. For every decision variable x_j and its input value $X_j \in X_B^{last}$
      - Add the goal constraint: $x_j + \gamma_j^- - \gamma_j^+ = X_j$
   c. - Add the goals F_α and bounds S_β of the opponent agent's (GP_α).[42]

2. Replace the objective-functions vector of (GP_β):
   a. Set the first priority function as the opponent's α objective functions.
   b. Set the rest of the priority functions according to the original β objective functions.
   c. Add at the bottom of the priority functions another level as $\sum_{j=1}^{m}\left(V_j^+\gamma_j^+ + V_j^- \cdot \gamma_j^-\right)$ with weight settings as in the other improve routines.
   d. Add at the bottom of the priority functions another level and put there a suitable Hanan function (see 0).

3. Call $gp\_solve(GP_\beta)$ and return its output.

---

[39] Note that in the cooperative algorithm we do not need the completed X_B^{last} tuple.

[40] The α_{Bi}^{last} values are taken from P_B^{last}.

[41] See Level-by-Level negotiation strategy.

[42] Both agents must agree on the decision variable representation.

Improve-function Initialization [43]

All the versions of the improve function (Ignorant, Aggressive, Cooperative) require as an input, the P_{agent}^{last}, which is the result of the agent's function last calculation that was presented already to the other party.

This section describes how to initialize this structure, i.e. how to calculate P_{agent}^{FIRST}[44].

Note that a proposal P may be required either by Sue or Bob (when the algorithm is informed). Hence, for clarity reasons, each proposal below is assigned an explicit agent, say B for Bob, or S for Sue, when concluding the initial proposal for the other agent should be trivial.

Improve-Ignorant

The algorithm, say from Bob's viewpoint, requires its last proposal P_B^{last} so that it can set goal constraints upon the region of its objective values, of the form:

$$\alpha_i \leq \alpha_{B(i)}^{last} + \rho \cdot \left(\alpha_{S(i)}^{last} - \alpha_i^*\right)$$

Initialization: $P_B^{FIRST} = P_B(\alpha^*)$, i.e., an optimal offer from Bob's point of view.

---

[43] Basically, this issue should be under the subject of the mechanisms that use the utility layer (and not utilities). However, we cover this issue here for completeness. Moreover, the initialization may be dependent upon the mechanism's strategy!

[44] Note that P_B^{FIRST} in the Sue's function, for example, consists of α_S^{FIRST} and the corresponding X tuple of values for the decision variables.

No special initialization is required here. The initial proposal will assign the objective-functions values as the optimal solution values, and the values for the decision-variables tuple are those in the optimal solution found.

Improve-Aggressive

The algorithm, say from Sue's viewpoint, requires Bob's P_B^{last} (45) so that it can set goal constraints upon the desirable values of the opponent's objective-functions values, of the form:

$$\alpha_i + \delta_i^- - \delta_i^+ = \alpha_{Si}^{last} - \rho \cdot \left(\alpha_{Si}^{last} - \alpha_i^*\right)$$

Initialization: $P_B^{FIRST} = P_B(\alpha_{\beta^*})$.[46]

That is, the initial proposal for Bob's objective values will be calculated according to the tuple X, which is generated when computing β^*.

Note that this sets α_S^{FIRST} as big as possible in β terms, and (in each iteration) the Improve function will decrease it towards α^*.

Improve-Cooperative

Sue's algorithm requires its P_S^{last} so that it can set goal constraints upon the region of its objective values, of the form:

$$\beta_{S(i)} \le \beta_{S(i)}^{last} + \rho \cdot \left(\beta_{S(i)}^{last} - \beta_i^*\right)$$

Initialization: $\beta_{S(i)}^{FIRST} = 1 + \kappa$ and $\kappa = 0.01 \cdot \left(\sum r.h.s / \# constraint\right)$

---

[45] Notice that this algorithm is informed, and it calculates P_B^{last} according to its information about Bob's problem,

When $\left(\sum r.h.s\right)$ is the sum of all the Right-Hand-Sides (r.h.s) of the original goals.

And $(\#constraint)$ is the number of the above goals.

Remarks:

1. The value of κ calculated above should be very close to the coefficient 0.01, this is due to the fact that all the goals are normalized, the (r.h.s $\leq$ 1).
2. The initialization of β_S^{FIRST} is concerned with pushing its value by a small amount from the value of β_i^*. Otherwise, the iterative function will be stuck with similar constraints of the form $\beta_{S(i)} \leq \beta_i^*$.

[ii] The expression $P_B\left(\alpha_{\beta^*}\right)$ means: the objective functions solution values of Bob (as they are represented at the informed Sue), when they are calculated according to the values of the Sue's optimal values β^*.

Using Hanan's method in the Improve Utilities

In the improve function, we build a new objective-functions list. Of course, we are interested in applying Hanan, on the agent's objectives, with the deviations that participated in the agent's original goal-constraints. This objective forms the least important objective layer. However, we note the following concerning the levels added to the original GP of the agent. There are three kinds of objective-levels added:

1. *X-close constraints (trying to keep close to the opposite offer):*
   All the deviation variables participating in the goal-constraints are considered, therefore, none of them can participate in the Hanan objective level.

2. *Objective-close constraints in the Aggressive and Ignorant functions:*
   All the deviation variables participating in the goal-constraints are considered, therefore, none of them can participate in the Hanan objective level.

3. *Opposite-agent constraints*

   In this case there seems no reason to apply Hanan on the deviations of the opposite-agent's GP problem. We should not be interested in generating non-inferior solutions using the opposite-agent's GP problem.

Advanced Utilities

Testing the necessity for negotiations

This is a pre-negotiation procedure whose objective is to test whether negotiation is needed. It is executed after unification and before one-to-one negotiations start.

1. Using the Suggest_optimal utility, solve for each agent (Bob and Sue) separately their goal programs and obtain their "independent" optimal solutions.
2. Construct a joint goal program whose set of constraints is the union of the two sets of constraints associated with the two agents plus

two additional goal constraints requiring that the two independent optimal values that were obtained earlier will hold. The objective of the joint program is to minimize the deviations associated with the last two constraints.

3. Solve the joint program. If its optimal value is zero we can skip negotiations, assign the optimal values of the joint program to the vector of decision variables (x) and guarantee that both agents can not achieve a better solution by any form of negotiations. If the optimal value of the joint program is not equal to zero, at least one of the agents may be able to obtain some gains through negotiations. In that case, we must enter negotiations.

Mediation option at any level

This utility can be called either before the negotiation were started or at any level during the negotiation. It is useful for several reasons:

- Applying the mediation option before negotiations were started provides the system with an "objective" outcome that can be compared against the outcome of negotiations later on. This will help various validation tests during development.
- If negotiations broke down at some intermediate level we might use mediation to "rescue" the deal. It will preserve the satisfaction levels achieved during the levels that were already finished successfully and generate a compromise solution for the level in which negotiations broke. This will strengthen our modeling foundations as it bypasses potential roadblocks on the way towards successful outcomes.
- From a marketing point of view, the users will appreciate the option to resort to a mediated solution if it is proven to be superior (for both sides) to the one they have achieved through negotiations.

Suppose we are now at level $k \geq 1$ of the objective function. The mediation option is then given by the following formulation.

$$\begin{array}{lll}
Min\_lex & (k) & \left\{\frac{\sum w_k^+ \cdot \delta_k^+ + \sum w_k^- \cdot \delta_k^-}{\sum w_k^+ + \sum w_k^-}\right\} + \left\{\frac{\sum v_k^+ \cdot \gamma_k^+ + \sum v_k^- \cdot \gamma_k^-}{\sum v_k^+ + \sum v_k^-}\right\} \\
 & (k+1) & \left\{\frac{\sum w_{k+1}^+ \cdot \delta_{k+1}^+ + \sum w_{k+1}^- \cdot \delta_{k+1}^-}{\sum w_{k+1}^+ + \sum w_{k+1}^-}\right\} + \left\{\frac{\sum v_{k+1}^+ \cdot \gamma_{k+1}^+ + \sum v_{k+1}^- \cdot \gamma_{k+1}^-}{\sum v_{k+1}^+ + \sum v_{k+1}^-}\right\} \\
 & \dots & \\
 & (last\ level\ q) & \left\{\frac{\sum w_q^+ \cdot \delta_q^+ + \sum w_q^- \cdot \delta_q^-}{\sum w_q^+ + \sum w_q^-}\right\} + \left\{\frac{\sum v_q^+ \cdot \gamma_q^+ + \sum v_q^- \cdot \gamma_q^-}{\sum v_q^+ + \sum v_q^-}\right\}
\end{array}$$

s.t.

Goal constraints of Bob

Goal constraints of Sue

Hard constraints of both Sue and Bob

Hard constraint to enforce satisfaction of objective levels 1.,,,. k-1 at the values that were achieved before

Form Offers

This procedure is concerned with forming a "compromise proposal" based on the GP's of two parties, Bob and Sue. Intuitively, it takes their preferences and constraints and finds a "middle ground" based on their targets. Using adjustable parameters, it may favor one party's interests over the other party's interests. This is superficially similar to the mediation procedure described earlier. The differences are that here we assume no prior negotiation session (and hence no achievements to preserve), in producing the mediated offer we do not normalize weights as previously done, and we "blur" all the levels whereas previously we followed the level structure. Hence, the current utility Form Offer is simpler and cruder than the previously described one.

Input

1. A tuple L of lower bounds for the decision variables.[47]
2. A tuple U of upper bounds for the decision variables.
3. Bob's *GP*: (GP_α)

   α – The problem objectives of Bob
   F_α – The goal constraints
   S_B – The hard-constraints
4. Sue's *GP*: (GP_β)

   β – The problem objectives of Sue
   F_β – The goal constraints
   S_S – The hard-constraints

---

[47] Note that both parties agree upon the bounds of the problem.

5. Bob's weight: ρ, implicitly Sue's weight is $(1-\rho)$.

Output

A tuple D, this is a proposed solution for the decision variables values.

Description

Given two players, $\{player^k\}$, $k \in \{1,2\}$, we denote Bob as k=1 and Sue as k=2. Solving the following problem provides the initial offer:

$$Min\_Lex \quad \rho \cdot \left\{ \sum_{j \in S^1} \left(\delta_j^- + \delta_j^+ \right) \right\} + (1-\rho) \cdot \left\{ \sum_{j \in S^2} \left(\delta_j^- + \delta_j^+ \right) \right\}$$

$$s.t.$$

$$\begin{aligned} & x_j + \delta_j^- - \delta_j^+ = T_j^1 && j \in player^1 \\ & x_j + \delta_j^- - \delta_j^+ = T_j^2 && j \in player^2 \\ & S_B, S_S && \\ & L_j \le x_j \le U_j && \forall j \end{aligned}$$

The program above is a "watered-down" version of the parties' Goal Programs. It considers only the bounds on individual variables and the targets of both players.

String-Final-Offer {Informed, Informed} (Sue's viewpoint)

See the Discrete variables compilation in section 4 of the compilation document.

Input:

1. Agent's *GP:* (GP_β)
    - β – Sue's objectives
    - F_β – Sue's goal constraints
    - S_S – Sue's hard-constraints
2. The opponent's GP:[48] (GP_α)
    - α – Bob's objectives
    - F_α – Bob's goal constraints
    - S_B – Bob's hard-constraints
3. A tuple X_S^{last} of values for the discrete-weight auxiliary variables that caused the agreement on the last level.
4. A tuple X_B^{last} of values for the discrete-weight auxiliary variables that caused the agreement on the last level.
5. Agent's agreement values for the objective function levels β_S^{agree}.
6. Opponent's agreement values for the objective function levels α_B^{agree}.

Note: The input GPs of this function must not contain ant level-by-level information. After all this is a mediation function, trying to find a suitable offer for both agents, hence, it is not wise to stick to the values already achieved.

Output:

A proposed solution P_S^{new}, which consists of:

---

[48] The opponent's information is subject to the mechanism's strategy. For example, the opponent may not provide all of his preferences (constraints or objectives), or even may not provide true preferences!

1. A *tuple* $T' = \left[D' \mid X' \right]$ of values for the variables. Especially the original discrete (string) decision variables.
2. A *tuple* β' of the corresponding objective-functions values.

Description:

This utility is used by the 1-1 negotiation mechanism to construct a mediation offer for the discrete (string) variables.

The negotiation should progress over the relaxed-binary variables, and once an agreement is achieved for the auxiliary variables (composed by the binary variables), this mediator function tries to find the closest offer that keeps the agreed vales and objectives.

The caller of this function is the one that decides to stop the negotiation, and agree to the opponent last offer; therefore that mediator gives more importance to the values of the opponent, expressed by bigger weights for the opponent deviation variables in the objective function.

1. Add to (GP_β) the following goal-constraints and bounds:
    a. Add the opponent's goal-constraints and bounds (GP_α)
    b. For every auxiliary variable x_{Si} and its input value $X_i \in X_S^{last}$
        - Add the goal constraint: $x_{Si} + \gamma_{Si}^- - \gamma_{Si}^+ = X_{Si}$
    c. For every decision variable x_{Bi} and its input value $X_{Bi} \in X_B^{last}$
        - Add the goal constraint: $x_{Bi} + \gamma_{Bi}^- - \gamma_{Bi}^+ = X_{Bi}$
    d. For every objective function-level of the agent β_i and its value β_{Si}^{agree}
        - Add the goal constraint: $\beta_i + \delta_{Si}^- - \delta_{Si}^+ = \beta_{Si}^{agree}$
    e. For every objective function-level of the opponent α_i and its value α_{Bi}^{agree}
        - Add the goal constraint: $\alpha_i + \delta_{Bi}^- - \delta_{Bi}^+ = \alpha_{Bi}^{agree}$
2. Replace the objective-function vector of (GP_α):
    a. Set the first priority level as:

$$1000 \cdot \sum \delta_{Bi}^+ + 100 \cdot \sum \delta_{Si}^+ + 10 \cdot \sum \left(\gamma_{Bi}^- + \gamma_{Bi}^+\right) + \sum \left(\gamma_{Si}^- + \gamma_{Si}^+\right)$$

3. Call $gp\_solve(GP_\beta)$ and return its output.

Notice that this is a mediation function therefore, the input GPs are the original agent's GPs, therefore the indexes β_S and α_B instead of the usual indexes β_S and α_S.

Second-price-bid {Informed, Ignorant} (Bob's viewpoint)

This function is presented from Bob's viewpoint, as it will be called Bobs participating in a second-price auction.

Input:

1. Agent's *GP*: (GP_β)
   - β – Sue's objectives
   - F_β – Sue's goal constraints
   - S_S – Sue's hard-constraints
2. The opponent's GP: (GP_α)
   - α – Bob's objectives
   - F_α – Bob's goal constraints
   - S_B – Bob's hard-constraints
3. Agent's private values for the objective function levels $\alpha_B^{private}$.
4. Opponent's minimum price values for the objective function levels β_B^{min}.

Output:

1. A proposed bid of the auctioneer objective values β'.

Description:

This utility is used by the 1-N negotiation mechanism (second-price auction) to construct a second-price bid, given the private value of the agent (Bob), and the minimum price of the auctioneer (Sue).

1. Add to (GP_α) the following goal-constraints and bounds:
   a. Add the opponent's goal-constraints and bounds (GP_β)

b. For every objective function-level of the agent α_i and its value α_{Bi}^{agree}

  - Add the goal constraint: $\alpha_i \leq \alpha_{Bi}^{agree}$

c. For every objective function-level of the opponent β_i and its value β_{Bi}^{agree}

  - Add the goal constraint: $\beta_i \leq \beta_{Bi}^{agree}$

2. Set the objective-functions vector for (GP_β) as of the opponent β_B.

3. Call *gp_solve* (GP_β) and return its output.

Note: The only concern of the function is to generate objective values, that's why the objective function is kept simple; E.g. we don't need the Hanan level, or to minimize the agent's own objective, as these will only affect the decision variables.

Improve-function Initialization[49]

All the versions of the improve function (Ignorant, Aggressive, Cooperative) require as an input, the P_{agent}^{last}, which is the result of the agent's function last calculation that was presented already to the other party.

This section describes how to initialize this structure, i.e. how to calculate P_{agent}^{FIRST}[50].

---

[49] Basically, this issue should be under the subject of the mechanisms that use the utility layer (and not utilities). However, we cover this issue here for completeness. Moreover, the initialization may be dependent upon the mechanism's strategy!

Note that a proposal P may be required either by Sue or Bob (when the algorithm is informed). Hence, for clarity reasons, each proposal below is assigned an explicit agent, say B for Bob, or S for Sue, when concluding the initial proposal for the other agent should be trivial.

Improve-Ignorant

The algorithm, say from Bob's viewpoint, requires its last proposal P_B^{last} so that it can set goal constraints upon the region of its objective values, of the form:

$$\alpha_i \le \alpha_{B(i)}^{last} + \rho \cdot \left(\alpha_{S(i)}^{last} - \alpha_i^*\right)$$

Initialization: $P_B^{FIRST} = P_B(\alpha^*)$, i.e., an optimal offer from Bob's point of view.

No special initialization is required here. The initial proposal will assign the objective-functions values as the optimal solution values, and the values for the decision-variables tuple are those in the optimal solution found.

---

[44] Note that P_B^{FIRST} in the Sue's function, for example, consists of α_S^{FIRST} and the corresponding X tuple of values for the decision variables.

Improve-Aggressive

The algorithm, say from Sue's viewpoint, requires Bob's P_B^{last} (51) so that it can set goal constraints upon the desirable values of the opponent's objective-functions values, of the form:

$$\alpha_i + \delta_i^- - \delta_i^+ = \alpha_{Si}^{last} - \rho \cdot \left(\alpha_{Si}^{last} - \alpha_i^*\right)$$

Initialization: $P_B^{FIRST} = P_B(\alpha_{\beta^*})$.[52]

That is, the initial proposal for Bob's objective values will be calculated according to the tuple X, which is generated when computing β^*.

Note that this sets α_S^{FIRST} as big as possible in β terms, and (in each iteration) the Improve function will decrease it towards α^*.

Improve-Cooperative

Sue's algorithm requires its P_S^{last} so that it can set goal constraints upon the region of its objective values, of the form:

$$\beta_{S(i)} \le \beta_{S(i)}^{last} + \rho \cdot \left(\beta_{S(i)}^{last} - \beta_i^*\right)$$

Initialization: $\beta_{S(i)}^{FIRST} = 1 + \kappa$ and $\kappa = 0.01 \cdot \left(\sum r.h.s / \# constra\,\mathrm{int}\right)$

When $\left(\sum r.h.s\right)$ is the sum of all the Right-Hand-Sides (r.h.s) of the original goals.

---

[51] Notice that this algorithm is informed, and it calculates P_B^{last} according to its information about Bob's problem.

[52] The expression $P_B(\alpha_{\beta^*})$ means: the objective functions solution values of Bob (as they are represented at the informed Sue), when they are calculated according to the values of the Sue's optimal values β^*.

And $(\#constra\mathrm{int})$ is the number of the above goals.

Remarks:

3. The value of κ calculated above should be very close to the coefficient 0.01, this is due to the fact that all the goals are normalized, the (r.h.s ≤ 1).

4. The initialization of β_S^{FIRST} is concerned with pushing its value by a small amount from the value of β_i^*. Otherwise, the iterative function will be stuck with similar constraints of the form $\beta_{S(i)} \leq \beta_i^*$.

Using Hanan's method in the Improve Utilities

In the improve function, we build a new objective-functions list. Of course, we are interested in applying Hanan, on the agent's objectives, with the deviations that participated in the agent's original goal-constraints. This objective forms the least important objective layer. However, we note the following concerning the levels added to the original GP of the agent. There are three kinds of objective-levels added:

4. *X-close constraints (trying to keep close to the opposite offer):* All the deviation variables participating in the goal-constraints are considered, therefore, none of them can participate in the Hanan objective level.

5. *Objective-close constraints in the Aggressive and Ignorant functions:* All the deviation variables participating in the goal-constraints are considered, therefore, none of them can participate in the Hanan objective level.

6. *Opposite-agent constraints*

In this case there seems no reason to apply Hanan on the deviations of the opposite-agent's GP problem. We should not be interested in generating non-inferior solutions using the opposite-agent's GP problem.

Advanced Utilities

Testing the necessity for negotiations

This is a pre-negotiation procedure whose objective is to test whether negotiation is needed. It is executed after unification and before one-to-one negotiations start.

1. Using the Suggest_optimal utility, solve for each agent (Bob and Sue) separately their goal programs and obtain their "independent" optimal solutions.

5.________Construct a joint goal program whose set of constraints is the union of the two sets of constraints associated with the two agents plus two additional goal constraints requiring that the two independent optimal values that were obtained earlier will hold. The objective of the joint program is to minimize the deviations associated with the last two constraints.

6.________Solve the joint program. If its optimal value is zero we can skip negotiations, assign the optimal values of the joint program to the vector of decision variables (x) and guarantee that both agents can not achieve a better solution by any form of negotiations. If the optimal value of the joint program is not equal to zero, at least one of the agents may be able to obtain some gains through negotiations. In that case, we must enter negotiations.

Mediation option at any level

This utility can be called either before the negotiation were started or at any level during the negotiation. It is useful for several reasons:

- Applying the mediation option before negotiations were started provides the system with an "objective" outcome that can be compared against the outcome of negotiations later on. This will help various validation tests during development.
- If negotiations broke down at some intermediate level we might use mediation to "rescue" the deal. It will preserve the satisfaction levels achieved during the levels that were already finished successfully and generate a compromise solution for the level in which negotiations broke. This will strengthen our modeling foundations as it bypasses potential roadblocks on the way towards successful outcomes.
- From a marketing point of view, the users will appreciate the option to resort to a mediated solution if it is proven to be superior (for both sides) to the one they have achieved through negotiations.

Suppose we are now at level $k \geq 1$ of the objective function. The mediation option is then given by the following formulation.

$$\begin{array}{lll}
Min\_lex & (k) & \left\{\dfrac{\sum w_k^{+} \cdot \delta_k^{+} + \sum w_k^{-} \cdot \delta_k^{-}}{\sum w_k^{+} + \sum w_k^{-}}\right\} + \left\{\dfrac{\sum v_k^{+} \cdot \gamma_k^{+} + \sum v_k^{-} \cdot \gamma_k^{-}}{\sum v_k^{+} + \sum v_k^{-}}\right\} \\
 & (k+1) & \left\{\dfrac{\sum w_{k+1}^{+} \cdot \delta_{k+1}^{+} + \sum w_{k+1}^{-} \cdot \delta_{k+1}^{-}}{\sum w_{k+1}^{+} + \sum w_{k+1}^{-}}\right\} + \left\{\dfrac{\sum v_{k+1}^{+} \cdot \gamma_{k+1}^{+} + \sum v_{k+1}^{-} \cdot \gamma_{k+1}^{-}}{\sum v_{k+1}^{+} + \sum v_{k+1}^{-}}\right\} \\
 & \cdots\cdot & \\
 & (last\ level\ q) & \left\{\dfrac{\sum w_q^{+} \cdot \delta_q^{+} + \sum w_q^{-} \cdot \delta_q^{-}}{\sum w_q^{+} + \sum w_q^{-}}\right\} + \left\{\dfrac{\sum v_q^{+} \cdot \gamma_q^{+} + \sum v_q^{-} \cdot \gamma_q}{\sum v_q^{+} + \sum v_q^{-}}\right\}
\end{array}$$

s.t.

Goal constraints of Bob

Goal constraints of Sue

Hard constraints of both Sue and Bob

Hard constraint to enforce satisfaction of objective levels 1,,,. k-1 at the values that were achieved before

Form Offers

This procedure is concerned with forming a "compromise proposal" based on the GP's of two parties, Bob and Sue. Intuitively, it takes their preferences and constraints and finds a "middle ground" based on their targets. Using adjustable parameters, it may favor one party's interests over the other party's interests. This is superficially similar to the mediation procedure described earlier. The differences are that here we assume no prior negotiation session (and hence no achievements to preserve), in producing the mediated offer we do not normalize weights as previously done, and we "blur" all the levels whereas previously we followed the level structure. Hence, the current utility Form Offer is simpler and cruder than the previously described one.

~~6.1.~~Input

6. A tuple L of lower bounds for the decision variables.[53]

7. A tuple U of upper bounds for the decision variables.

8. Bob's GP: (GP_α)

α – The problem objectives of Bob

F_α – The goal constraints

S_B – The hard-constraints

9. Sue's GP: (GP_β)

β – The problem objectives of Sue

F_β– The goal constraints

S_S – The hard-constraints

10. Bob's weight: ρ, implicitly Sue's weight is $(1-\rho)$.

Output

A tuple D, this is a proposed solution for the decision variables values.

Description

Given two players, $\{player^k\}$, $k \in \{1,2\}$, we denote Bob as k=1 and Sue as k=2. Solving the following problem provides the initial offer:

---

[53] Note that both parties agree upon the bounds of the problem.

$$Min\_Lex \qquad \rho \cdot \left\{ \sum_{j \in S^1} \left(\delta_j^- + \delta_j^+ \right) \right\} + (1-\rho) \cdot \left\{ \sum_{j \in S^2} \left(\delta_j^- + \delta_j^+ \right) \right\}$$

s.t.

$$\begin{aligned}
& x_j + \delta_j^- - \delta_j^+ = T_j^1 && j \in player^1 \\
& x_j + \delta_j^- - \delta_j^+ = T_j^2 && j \in player^2 \\
& S_B, S_S && \\
& L_j \le x_j \le U_j && \forall j
\end{aligned}$$

The program above is a “watered-down” version of the parties’ Goal Programs. It considers only the bounds on individual variables and the targets of both players.

String-Final-Offer {Informed, Informed} (Sue's viewpoint)

See the Discrete variables compilation in section 4 of the compilation document.

Input:

7. Agent's GP: (GP_β)

β – Sue's objectives
F_β – Sue's goal constraints
S_S – Sue's hard-constraints

8. The opponent's GP:[54] (GP_α)

α – Bob's objectives
F_α – Bob's goal constraints
S_B – Bob's hard-constraints

9. A tuple X_S^{last} of values for the discrete-weight auxiliary variables that caused the agreement on the last level.

10. A tuple X_B^{last} of values for the discrete-weight auxiliary variables that caused the agreement on the last level.

11. Agent's agreement values for the objective function levels β_S^{agree}.

12. Opponent's agreement values for the objective function levels α_B^{agree}.

Note: The input GPs of this function must not contain ant level-by-level information. After all this is a mediation function, trying to find a suitable offer for both agents, hence, it is not wise to stick to the values already achieved.

---

[54] The opponent's information is subject to the mechanism's strategy. For example, the opponent may not provide all of his preferences (constraints or objectives), or even may not provide true preferences!

Output:

A proposed solution P_S^{new}, which consists of:

3. A *tuple* $T' = \left[D' \mid X'\right]$ of values for the variables. Especially the original discrete (string) decision variables.

4. A *tuple* β' of the corresponding objective-functions values.

~~5.3.~~Description:

This utility is used by the 1-1 negotiation mechanism to construct a mediation offer for the discrete (string) variables.

The negotiation should progress over the relaxed-binary variables, and once an agreement is achieved for the auxiliary variables (composed by the binary variables), this mediator function tries to find the closest offer that keeps the agreed vales and objectives.

The caller of this function is the one that decides to stop the negotiation, and agree to the opponent last offer; therefore that mediator gives more importance to the values of the opponent, expressed by bigger weights for the opponent deviation variables in the objective function.

4. Add to (GP_β) the following goal-constraints and bounds:

   f. Add the opponent's goal-constraints and bounds (GP_α)

   g. For every auxiliary variable x_{Si} and its input value $X_i \in X_S^{last}$

   - Add the goal constraint: $x_{Si} + \gamma_{Si}^{-} - \gamma_{Si}^{+} = X_{Si}$

   h. For every decision variable x_{Bi} and its input value $X_{Bi} \in X_B^{last}$

   - Add the goal constraint: $x_{Bi} + \gamma_{Bi}^{-} - \gamma_{Bi}^{+} = X_{Bi}$

i. For every objective function-level of the agent β_i and its value β_{Si}^{agree}

- Add the goal constraint: $\beta_i + \delta_{Si}^- - \delta_{Si}^+ = \beta_{Si}^{agree}$

j. For every objective function-level of the opponent α_i and its value α_{Bi}^{agree}

- Add the goal constraint: $\alpha_i + \delta_{Bi}^- - \delta_{Bi}^+ = \alpha_{Bi}^{agree}$

5. Replace the objective-function vector of (GP_α):

~~a.~~b. Set the first priority level as:

$$1000 \cdot \sum \delta_{Bi}^+ + 100 \cdot \sum \delta_{Si}^+ + 10 \cdot \sum \left(\gamma_{Bi}^- + \gamma_{Bi}^+\right) + \sum \left(\gamma_{Si}^- + \gamma_{Si}^+\right)$$

6. Call *gp_solve*(GP_β) and return its output.

Notice that this is a mediation function therefore, the input GPs are the original agent's GPs, therefore the indexes β_S and α_B instead of the usual indexes β_S and α_S.

~~6.~~Second-price-bid {Informed, Ignorant} (Bob's viewpoint)

This function is presented from Bob's viewpoint, as it will be called Bobs participating in a second-price auction.

Input:

5. Agent's *GP:* (GP_β)

β – Sue's objectives
F_β – Sue's goal constraints
S_S – Sue's hard-constraints

6. The opponent's GP*:* (GP_α)

α – Bob's objectives
F_α – Bob's goal constraints
S_B – Bob's hard-constraints

7. Agent's private values for the objective function levels $\alpha_B^{privatee}$.

8. Opponent's minimum price values for the objective function levels β_B^{min}.

Output:

2. A proposed bid of the auctioneer objective values β'.

Description:

This utility is used by the 1-N negotiation mechanism (second-price auction) to construct a second-price bid, given the private value of the agent (Bob), and the minimum price of the auctioneer (Sue).

4. Add to (GP_α) the following goal-constraints and bounds:

a. Add the opponent's goal-constraints and bounds (GP_β)

b. For every objective function-level of the agent α_i and its value α_{Bi}^{agree}

- Add the goal constraint: $\alpha_i \le \alpha_{Bi}^{agree}$

c. For every objective function-level of the opponent β_i and its value β_{Bi}^{agree}

- Add the goal constraint: $\beta_i \le \beta_{Bi}^{agree}$

5. Set the objective-functions vector for (GP_β) as of the opponent β_B.

6. Call $gp\_solve(GP_\beta)$ and return its output.

Note: The only concern of the function is to generate objective values, that's why the objective function is kept simple; E.g. we don't need the Hanan level, or to minimize the agent's own objective, as these will only affect the decision variables.

Catalog Purchasing of a Single Item

Introduction

This chapter addresses the special case of negotiations which are held, either in a 1-1 or 1-N settings, between buyer(s) and a supplier who sells items out of a catalog. The catalog is organized by families (or groups) of items (e.g., a car catalog organized by 3 groups: luxury, sedan, 4X4). Within each family, items are listed according to their attributes. The attributes are typically given in a discrete fashion (e.g., engine size may be limited to a vector of 4 values {1600, 1800, 2000, 2400}). Each item in the catalog represents a specific vector of attribute values that are permissible for sale from the point of view of the seller. For example, item no. x in

the catalog is defined by
{engine size = 1800}, {color = white}, {model = GLX}, {list price = \$16000}.

Not all the combinations of attribute values are feasible in the catalog. Thus, for example, the combination
{engine (E) = 1800}, {color (C) = silver}, {model (M) = GLX}, {price (P) = \$16000} is not available although there are other cars in the catalog whose color is silver.

We address several scenarios that are distinguished according to the following characteristics:

- Access to catalog
    - The catalog was published and every potential buyer has access to it
    - The supplier is the only one who knows the contents of the catalog
- Knowledge of buyer's intention
    - The seller knows the GP according to which the buyer evaluates offers
    - The seller is ignorant of the buyer's GP
- Number of relevant items
    - There are a few relevant items
    - There are possibly many relevant items.

In all cases we assume that the unification procedure is capable of generating the appropriate upper and lower bounds on the various attribute dimensions (e.g., $1600 \leq E \leq 2400$). The negotiation procedure will be composed of two stages. In the first stage, negotiations will be held along the relevant feasible intervals of the various attributes without regard to whether or not an offer corresponds to an item that actually exists in the catalog. At the second stage, the seller's intention will be duplicated to a number of intentions that will run in the system in parallel where each intention corresponds to a particular item in the catalog. The motivation for the first step is to limit the number of items on which negotiation will eventually take place. Since it is reasonable to think that in most practical cases the catalog will consist of a large number of items we want to avoid representing each catalog item by an integer variable (which will result in a large integer programming problem).

To facilitate the execution of these two stages we require two additional parameters:

K: a threshold number of candidate items. Once that number is reached, the seller duplicates its original intention into K separate intentions, each corresponding exactly to a particular item in the catalog. All K sub-intentions continue into the second stage where they are engaged in parallel negotiations with the buyer's intention and maintain an OR condition among them. The value of K should be fairly small (say, about 10) to scalability problems. Also, in case the buyer is operating in a manual mode of negotiations, K should be even smaller (say, not exceeding 5).

S: a subset of candidate items out of the complete catalog. This subset is defined differently according to the possible scenarios. If the seller is knowledgeable, then S is defined according to the last offer made by the seller as it is evaluated by the buyer's objective function. All the catalog items whose attributes would lead to improvement from the buyer's standpoint are in S. If the seller is ignorant, S is defined by a certain proportion by which the seller is ready to worsen his offers. All items whose values, according to the seller's own objective, are not worse than the current seller's offer by more than the given proportion are in S.

Negotiation procedure

Stage 1

Buyer and seller exchange offers that are generated through the ordinary GP procedures without regard to whether or not there are items in the catalog that actually correspond to these offers. In each iteration, before sending his counter-offer, the seller finds the item in the catalog whose GP value is the closest to the one he has just computed. If he is ignorant, then he looks for the items whose values are closet from the perspective of his own GP. Otherwise, he looks for items that are closest from the perspective of the buyer's GP. Also, in each iteration the seller checks whether the number of items in S is still bigger than K. If not, he initiates a switch into the second stage. Computationally, the testing of how many items are in S is performed as follows:

- If the seller is knowledgeable he uses the buyer's GP to pre-process the entire catalog. Each item gets a "score" according to the buyer's objective and these scores are then used to rank order the items in a descending order of utility to the buyer. Given the value of the seller's current offer according to the buyer's objective function, all items whose rank position is smaller than or equal to the item with the closest value to the one computed are in S.

- If the seller is ignorant he uses his own GP to pre-process the entire catalog. Each item receives a score and the scores are used to rank order the items in an ascending order of their utility to the seller. Given the value of the buyer's current offer (from the seller's perspective), all the items whose rank positions are smaller than or equal to the item with the closest value to the one computed for the current buyer's offer are in S.

Stage 2

This stage can be executed in two ways.

a. The seller duplicates his original intention into K duplicates that maintain an OR relations as they enter into parallel 1-1 negotiations with the buyer's single intention. Each of the duplicate intentions corresponds to a particular item in the catalog – that is, in each such intention the variables that describe the item are fixed according to the relevant item from the catalog. For example, engine size, color and model type are fixed and negotiation can continue on variables that were not fixed such as price, warranty period, delivery date, etc.

b. The seller formulates a single GP that uses integer variables to define the K specific alternatives that are relevant to the negotiation.

This approach is somewhat more difficult to run due to the presence of integer variables but, on the other hand, it doesn't require an external procedure that verifies the OR relations as in the first approach. Another advantage here (from the seller's standpoint) is the extra flexibility to move among the catalog items during the negotiation – a possibility that doesn't exist in the first approach.

Reference is now made to Fig. 29, which is a simplified flow diagram which further explains the procedure from the seller's standpoint. In Fig. 29, the procedure is shown in three stages, pre-processing, stage 1 and stage 2.

Stage 2:

Example

The buyer's intention:

At importance level no. 1 the buyer is interested in price (as low as possible) and warranty (as high as possible). At importance level no. 2, his preferred engine size is 1800, deviations below it are 4 times worse than the other way around. The buyer has no preference on color. His intention is then formulated as the following GP:

$$\begin{array}{lll} Min\_Lex & \{\delta_W^- + \delta_P^+\}, \{4 \cdot \delta_E^- + \delta_E^+\} & \\ s.t. & & \\ & \frac{E}{800} + \delta_E^- - \delta_E^+ = \frac{1800}{800} \quad , & 1600 \le E \le 2400 \\ & \frac{P}{60000} + \delta_P^- - \delta_P^+ = \frac{50000}{60000} \quad , & 50000 \le P \le 110000 \\ & \frac{W}{5} + \delta_W^- - \delta_W^+ = \frac{5}{5} \quad , & 1 \le W \le 5 \end{array}$$

Notice that scaling of the goal constraints is done by their intervals (both here and later on for the seller). The buyer's GP stays the same as we switch from stage 1 to stage 2 (in fact, the switch is transparent to him).

The seller's intention:

Assume that the items in the seller's catalog are given in the following table:

Item	Engine	Warranty	Color	Manufacturing cost	Tar price

				cost	price
1	1600	1	Blue	63000	760
2	1600	1	Red	62500	750
3	1600	1	White	62000	740
4	1800	3	White	74500	890
5	1800	3	Blue	75000	900
6	2000	4	Silver	93000	110
7	2000	4	White	90000	108

As his first importance level the seller desires to maximize his profits and at the second level he wishes to minimize the warranty period. The seller is willing to move only one year above or below its catalog definition of the warranty period. Notice that in this example:

- The seller's main decision variable is profit. The buyer is totally unaware of this variable as well as the "manufacturing cost" column that was used to generate it.
- The largest profit is not necessarily associated with the item whose target price is the largest.

The, the seller's GP in the first stage would look like this:

$$Min\_Lex \qquad \{\delta_R^-\},\{\delta_W^+\}$$

$$s.t.$$

$$\frac{E}{400}+\delta_E^- -\delta_E^+ =\frac{2000}{400}$$

$$\frac{W}{5}+\delta_W^- -\delta_W^+ =\frac{1}{5}$$

$$\frac{P-90000}{6000}+\delta_P^- -\delta_P^+ =\frac{18000}{6000}$$

Suppose that k=3 and that the seller's first offer was car no. 7 (with list price of 108000). Then, S was composed of all the cars in the catalog. Let's say that after some iterations the seller's GP led to a hypothetical car {E=1600, C=silver, W=2, P=75800}. The actual car closest to it is car no. 1. At this point S is composed of only 3 items and the seller switches to the second stage.

Approach a: To formulate the seller's GP in terms of the discrete options that are available in his catalog we need to define a set of binary variables (say, X_i) where each such variable indicates the selection of a particular item (car) in the catalog. In this case we write:

$$
\begin{aligned}
&Min\_Lex \quad \{\delta_R^-\},\{\delta_W^+\}\\
&s.t.\\
&E = 1600\cdot(X_1+X_2+X_3)+1800\cdot(X_4+X_5)+2000\cdot(X_6+X_7)\\
&C = Blue\cdot(X_1+X_5)+\mathrm{Re}\,d\cdot(X_2)+White\cdot(X_3+X_4+X_7)+Silver\cdot(X_6)\\
&W+\delta_W^- -\delta_W^+ = 1\cdot(X_1+X_2+X_3)+3\cdot(X_4+X_5)+4\cdot(X_6+X_7)\\
&P+\delta_P^- -\delta_P^+ = 76000\cdot X_1+75000\cdot X_2+\ldots+108000\cdot X_7\\
&\frac{P-76000\cdot X_1+75000\cdot X_2+\ldots+108000\cdot X_7}{6000} \qquad +\delta_R^- -\delta_R^+ = \frac{18000}{6000}\\
&\sum_i X_i = 1 \quad , \quad X_i \in \{0,1\}, \forall i\\
&0 \le \delta_W^-, \delta_W^+ \le 1
\end{aligned}
$$

Notice that in this formulation deviations in engine size and color are not allowed for any of the seven possible items. These two represent fixed dimensions in the item's attribute space that cannot be changed by negotiations.

Approach b: The seller constructs 3 sub-intentions (one for each of the cars with E=1600) and let them run in parallel 1-1 sessions. For example, the GP that corresponds to item no. 1 in the catalog will then look like:

$$
\begin{aligned}
&Min\_Lex \quad \{\delta_R^-\},\{\delta_W^+\}\\
&s.t.\\
&E = 1600\\
&C = Blue\\
&\frac{W}{5}+\delta_W^- -\delta_W^+ = \frac{1}{5}\\
&\frac{P}{60000}+\delta_P^- -\delta_P^+ = \frac{76000}{60000}\\
&\frac{P-63000}{6000}+\delta_R^- -\delta_R^+ = \frac{13000}{6000}
\end{aligned}
$$

Implementation Comments

1. There is a need to find an efficient way to perform the pre-processing stage as it requires an exhaustive computation effort across all items in the catalog. Hence, when moving from one item in the catalog to its neighbor we might use the previous solution as the starting solution for the next run of the "Complete-Optimal" utility (see the Utilities chapter).

2. Finding the "closest" actual item requires a utility that will be based on the ranking achieved in the pre-processing stage. It does NOT require re-running one of the optimization utilities.

Catalog purchasing of multiple items

Introduction

This section extends the single item procedures to address complex deals that are composed of several items that need to be purchased simultaneously from (possibly) several catalogs. Here are some examples:

- A computer system including a PC, terminal and a printer
- A car with a trailer (we will refer to this example in the remainder of this document).
- A travel package including flight ticket and hotel reservation.

Within each intention we assume that there exists a natural ordering of the items according to their importance (e.g., a car is more important than a trailer). If some items are equally important, an arbitrary order can be established among them. We distinguish between two basic cases (the reminder of this document will address the second case):

1. Independent sub-intentions – There are no constraints of any kind (e.g., trade-offs, hard constraints) that tie the sub-intentions to one another. In such cases, each sub-intention can be dealt with according to the procedure outlined for single-item purchasing.
2. Dependent sub-intentions – There exist some constraints link the sub-intentions. For example, due to differences in standards, trailers of type A can be mounted only on European made cars while trailers of type B can be mounted only on American made cars. The relevant constraint will be:
   Trailer_hook_type = Car_hook_type

The problem statement

In a multi-item deal there may be many combinations of items (from a collection of catalogs) that may satisfy the constraints. Treating each one of these possibilities separately is often infeasible. An added complication is that if we generate an intention per combination, we lose significant bargaining power since in each negotiation session we deal with exactly one such item combination and it is not possible to move from one combination to another. Of course, a large number of parallel negotiation sessions is infeasible when one of the parties works manually.

The proposed solution

Unification Stage:

We perform the unification stage under the optimistic assumption that, within the given ranges of values for the relevant attributes, all possible items exist in their respective catalogs. Therefore, we actually perform no catalog searches during unification. At the conclusion of the unification stage we have an offer for which there might exist many possible combinations of specific catalog items. Let I be the joint intention at this point. Referring to our previous example, we describe the process assuming that intention I includes two items, a car and a trailer. The car (respectively, trailer) component in I is a hypothetical car (respectively, a hypothetical trailer). For example, I might be defined by:

Hypothetical Car: {Color = red, 1600 < Engine < 3000 cc, 60000 < Price < 120000}

Hypothetical Trailer: {80 < weight < 300, 0 < height < 65 cm, 0 < capacity < 2.5 ton}

Pre-processing Stage:

We compile a list of potentially feasible cars (C_{S0}) by enacting the following query. Find the list of cars that could possibly unify with the hypothetical car in I and for which there exists at least one "matching" trailer (effectively computing a semijoin of trailers into cars, satisfying all the relevant GP constraints on cars and trailers). Thus, for the example above, any car whose color is not red will not enter C_{S0}. We grade each car in C_{S0} according to the highest possible grade (least penalty) it could achieve using the seller's GP. For this grading we assume that any trailer we may want is available. We sort the list C_{S0} from best to worst. Next, we compile a list of potentially feasible trailers (T_{S0}) by employing a similar process for the trailers. In case the buyer's GP is known, we compute two similar lists (C_{B0} and T_{B0} for the cars and trailers, respectively) using the buyer's GP. In case the buyer's GP is not available, we just consider the relevant trailers as the semijoin of cars into trailers. Returning to the example above, this stage may lead to four lists as demonstrated below.

C_{S0}	No.	1	2	...	12	13	
	Car sku no.	17	122	...	33	177	
	Seller's value	25	28	...	45	48	

T_{S0}	No.	1	2	...	26	27	..
	Trailer sku no.	111	27	...	22	26	..
	Seller's value	5	12	...	42	49	..

C_{B0}	No.	1	2	...	21	22	..
	Car sku no.	33	48	...	167	93	..
	Buyer's value	12	16	...	39	52	..

T_{B0}	No.	1	2	...	9	10	...
	Trailer sku no.	27	88	...	67	21	...
	Buyer's value	8	19	...	37	59	...

Negotiation Stages - Introduction

Negotiation proceeds in two stages. In stage 1 the seller operates with hypothetical items when offers are computed. However, when the seller presents an offer to the buyer, a concrete offer, based on actual catalog items, is presented. We shall explain how to locate such a concrete set of items for such a concrete offer. In stage 2 the seller only operates with concrete items. This is done either by (1) duplicating intention I, at this point, for each particular combination, or (2) employing an integer programming formulation that explicitly represents the choices still possible for the intention. The passage from stage 1 to stage 2 is performed when stage 1 fails to identify a combination that will meet its self-imposed requirements.

Negotiation Stage 1

(1) Suppose the seller has to respond to the buyer's last offer (given by the vector x). We distinguish between "non-negotiable" and "negotiable" attributes in the catalogs. For a given car, attributes such as color and engine size are non-negotiable while price and warranty period are negotiable. The seller first tries to generate a counter offer by operating an appropriate utility ("knowledgeable" when he knows the buyer's GP and "ignorant" otherwise) with the additional restriction that changes are allowed only in negotiable attributes. Only when this is not feasible, he employs the following procedure (he will also use this procedure when he has to provide an initial offer).

(2) First, assume the buyer's GP is known. The seller produces a hypothetical offer x using the "knowledgeable" utility with no additional restrictions. Denote the value for x to the seller as v and its value to the buyer's as w. Intersect the sets C_{S1} and C_{B1}, where C_{S1} contains all the cars in C_{S0} whose values for the seller are not worse by more than u% than v, and C_{B1} contains all cars in C_{B0} whose values for the buyer are not worse by more than q% than w. Both u and q are parameters. A similar computation is performed to find the trailer sets T_{S1} and T_{B1}. Let C_1 and T_1 be the resulting intersections, where C_1 is a set of cars and T_1 is a set of trailers. In the context of our example, suppose v = 46.5 and w = 40.4 and let u = q = 3.5%. Then, C_{S1} will contain the first 13 cars in C_{S0} and C_{B1} will contain the first 21 cars in C_{B0}. The cardinality of the intersection set C_1 will be smaller than or equal to 13 and it will contain at least the car whose sku no. is 33.

Next, consider the case in which the buyer's GP is not known. In this case, $C_I = C_{SI}$ and $T_I = T_{SI}$ (notice that these sets contain the sets C_I and T_I as prescribed above). As we shall see, in this case we may need to work harder at finding a combination with the desired properties.

Now, the seller needs to perform a sequential scanning of combinations of items from C_I and T_I until he either finds a valid combination whose value is "close enough to w" or until he finds that no such combination exists. Offers are generated only for valid combinations of items (car-trailer in our example; we assume that cars are more important than trailers).

(i) Order the cars in C_I from worst to best in terms of their value to the buyer, select the first car and denote it as Candidate_Car

(ii) Order the trailers in T_I from worst to best in terms of their value to the buyer, select the first trailer and denote it as Candidate_Trailer

(iii) Assign the values of both Candidate_Car and Candidate_Tar into the GP of I. If the GP is feasible and its value is not better by more than p% (p is a paremeter) than w, then go to (vii)

(iv) Else, make the next trailer in T_I the Candidate_Trailer and return to (iii)

(v) If T_I was exhausted, make the next car in C_I the Candidate_Car and return to (ii)

(vi) If C_I was exhausted, move to stage 2 of the negotiations

(vii) Generate offer

Negotiation Stage 2

The seller needs to backtrack one step to the previous instance of stage 1. There, instead of finding the FIRST combination that meets his self-imposed restrictions, he needs to perform an exhaustive scan to find ALL such combinations. We naturally assume that this will be a fairly small number since in the next iteration NO such combination was found. After finding all the combinations that meet the requirements, the seller either duplicates I according to each car-trailer combination or formulates a single integer-programming GP model (see the explanation for the single item catalog purchasing) to continue the negotiations.

Buyer's operation:

There are two basic possibilities for the buyer, namely knowledgeable and ignorant (of the seller's GP). These possibilities dictate how the buyer computes his offers, as usual. Additionally, there are two sub-cases, one in which the buyer has access to the catalog and one in which he does not. In the latter sub-case, the buyer works in terms of hypothetical cars. In the first sub-case, the buyer may still choose to work with hypothetical cars (and leave the seller the responsibility to come up with concrete ones); alternatively, the buyer may choose to work with concrete cars as well. In that case the buyer's situation is analogous to that described previously for the seller. He too may maintain lists and derive appropriate combinations. This option is costly and as default the buyer will work with hypothetical cars, unless he explicitly indicates willingness to employ the slower and more expensive option of locating concrete combinations of items for each offer. Should the buyer choose to work manually, he may choose items from the catalog and the system may assist him in grading the combinations.

Background

This section focuses on negotiation technologies for multi-attribute, multi-items deals where parties have constraints, preferences and trade-offs. Here, parties may be human participants, devices, or in general any sort of agent. The technology is applicable to eCommerce, resource allocation and interpersonal or inter-organizational negotiations. We cover:

- 1-N negotiations (auctions and reverse auctions)
- 1-1 (human-like) negotiations
- Profiles for 1-1 negotiations

We categorize the handling of intentions into the following logical layers:

- Building intentions, (done at the graphical user interface GUI) level; this level may correspond to both humans and automatic tools.

- The Unification of intentions (which uses the utility level).
- The Mechanism layer which can implement 1-1, 1-N mechanisms.
- The Utility level that handles various optimization and basic negotiation functionalities.

A schematic illustration of the architecture is shown in Fig. 30, which shows the various layers as described above.

Centralized and Distributed Systems

The description in this section is for the most part in terms of a centralized system. However, the techniques here are applicable to the case of a distributed system in which the negotiating parties either have a local copy of the system, each with its own layers (e.g., utility layer) and they communicate via messages, or they send messages to a central site where a single negotiating system is in operation. There are cases in which knowledge of another party's data is needed; in this case we assume that the relevant party sends it.

The One-to-N Mechanism

Introduction

We present procedures for markets characterized by a single agent (e.g., seller, buyer, barterer) operating against multiple agents (e.g., buyers, sellers, barterers) where the goal is to maximize the single agent's revenue from the deal. When the single agent is a buyer (respectively, seller) and the multiple agents are sellers (respectively, buyers), this is called an auction (respectively, reverse auction).

We start by defining the private value and interdependent value models. For simplicity we first assume that the only variable left to agree upon is the price or price-like (that is, a well defined domain in which all agents assign identical values to all points in the domain and where there is a total order on the domain). Later on, the relaxation of this assumption will be discussed.

In the private value setup, the values that an agent assigns to the different possible deals do not depend on information provided by other agents. Putting it differently, the agent's values of the different deals depend only on his own information. An example of such deals might be the purchase of a certain component by a purchasing officer. He knows the price of the final product (P), the cost of labor (L), and the margin required by the firm (M). These parameters determine for him an upper bound on the cost of the part (C) given through: $C \leq P - L - M$. Regardless of what other buyers in this market are willing to pay our purchasing officer will fix his private value according to C.

In interdependent value model, an agent's value depends not only on his own assessment but also on the information other agents possess. An important aspect of this model is that agents not only do not know each other's valuation, they also do not know their own value. An example of such a deal is an auction of land for oil drilling. In this example each agent performs some tests to assess the amount and quality of oil in the ground. Clearly, having the results of all tests refine ones own assessment.

The Private Value Model

We state a known result concerning price only auctions. If buyers' values are private, and every buyer's value is independently drawn from the same distribution, then the famous "Revenue Equivalence" theorem holds. This theorem states that all auctions in which the bidder with the highest value wins, and the one with the lowest value loses and pays zero, are equivalent from the seller's point of view. Note that the procedure we present below, which is a second price auction with a reservation price, does not belong to this class, and hence the Theorem is not applicable (this is because we allow the seller to announce a positive reservation price which might result with no trade).

A second price auction with a reservation price: Price and Price-like

Second price auctions are well known. The rules are as follows:

- Seller chooses a reservation price r, *which is then revealed,* to all buyers.

- Bidders simultaneously submit bids. We denote by p_j the bid of buyer j, $j=1,2,...,N$. (The buyers do not have to bid simultaneously but it is important that no buyer sees the bid of the others before he submits his bid).
- If all bids are below the seller's reservation price, then the procedure is terminated with no trade. Otherwise,
    - The winner is the bidder whose bid was the highest (if there is a tie then the winner is chosen by a random device).
    - If buyer *k* is the winner then he pays Max$[max_{j \in k}(p_j), r]$ to the seller and the other buyers pay zero.

Remarks:

1) At first blush, such an auction procedure looks inferior from a revenue point of view when compared, say, with the *first price auction* (a similar procedure except that the winner pays his own price and not the second). However, such a naive view ignores the effect of the procedure's type on how competitive the bidders are.

2) An important advantage of this procedure is that it is very simple and transparent to follow. When the bidders and the auctioneer argue on the same variables a dominant strategy for a buyer is to bid his value! That is, regardless of what other buyers intend to do, bidding one's own value is always optimal. However, notice that in multi-dimensional settings the bidders may be concerned with different variables than those that are of interest to the auctioneer.

3) For a seller to choose a positive reservation price makes sense only if it is a credible threat to terminate the procedure without a trade. If such a threat is not credible, then buyers are going to take it into account, and adjust their bids. While choosing the right bid is a simple matter in such a set up, choosing the optimal reservation price is a more difficult task. In particular, the optimal reservation price depends on the seller's belief on how the buyers' values are distributed.

4) We described an auction. However, a similar description would work for reverse auctions in which the auctioneer is a buyer, the bidders are sellers and the reservation price is the maximum value the seller is ready to

consider. Of course, the lowest bidder wins here and Max is replaced by Min in determining the deal's actual price.

Multi-dimensional Second Price Auction

If the only detail yet to agree upon is price (or price-like), then all sides understand the preferences of the other sides. Sellers are interested in a higher price and buyers prefer lower prices. This is not the case when we have multi-dimensional deals (i.e., deals that are evaluated based on parameters that are not necessarily price). In this case we denote a party's value function by *f(.)* or *g(.), possibly with subscripts.* Without loss of generality, and for the sake of consistency with our conventional representation of objective functions in the utility layer, we shall treat both *f(.)* and *g(.)* as functions that represent penalties. That is, the lower it is for a party, the better is for that party. We also observe the following: we no longer refer here to buyer or seller as each party evaluates the deal according to his function (this provides a uniform treatment for auctions, reverse auctions, bartering auctions etc.).

In such a case, it is important to first reveal the value function *g(.)* of the auctioneer to the bidders. One possibility is to reveal to the bidders the true value function *g(), which* was submitted by the auctioneer. Another possibility is to give the auctioneer an option to reveal a modified value function, denoted by *g'(.)*, according to his strategic choice. Without loss of generality let *g'(.)* be the disclosed value function.

So, first the auctioneer reveals his value function *g'(.)* (by publishing his GP_A: constraints, preferences and targets, arranged in K levels in lexicographical order) and then the bidders bid. Let

- The auctioneer's reservation price is given in terms of maximal allowed values for his objective function levels -- g_k. (A reasonable variation here is just to provide the first level.)
- $f_i(.)$ be the value function of bidder i, is expressed via goal program GP_i.
- f_{ih} are maximal allowed values for bidder *i*'s objective function, which are arranged in H_i levels, in lexicographical order.
- $b=(b_1,\dots b_n)$ denotes the vector of submitted bids, where each bid is a vector of values for the relevant decision variables (attributes). (A

reasonable alternatives is to just supply the value of *g'(.)*, see remark below).

Then, to generate a bid, bidder i solves a program that maximizes the seller's utility (by minimizing deviations from the seller's targets) subject to his own constraints as well as those of the auctioneer (i.e., both GP_i and GP_A) and while not crossing the threshold values specified for his own objective function levels as well as the thresholds that were specified by the seller:

$$\begin{array}{lll} Lex\_Min & \{\beta_1\},\{\beta_2\},\ldots,\{\beta_k\} & \\ s.t. & & \\ & \alpha_h \le f_{ih} & h = 1,\ldots,H_i \\ & \beta_k \le g_k & k = 1,\ldots,K \\ & GP_i & \\ & GP_A & \end{array}$$

The first set of hard constraints forces the solution to meet the bidder's threshold specifications (f_{ih}) at each level of his objective function. The second set of constraints restricts the bid so that it gives the auctioneer at least what he announced as his threshold levels g_k. At the same time, these constraints define the deviation variables that are used in the objective function. In the objective function of the program above, the bidder tries to respond, as best as he can under the constraints, to the auctioneer's objectives (according to their lexicographical order).

The auctioneer selects the winning bid as the one associated with the smallest value for *g'(.)*. Let bidder *m* be the winner. That is

$b_m = argmin_i[g'(b_i)]$

Let *s* denote the second winning bidder. That is,

$b_s = argmin_{i \ne m} [g'(b_i)]$

The traded deal, denoted by *d, then* solves

$$\begin{array}{ll} Lex\_Min & \alpha_m(d) \\ s.t. & \\ & \beta_A \le g'(b_s) \\ & GP_m \\ & GP_A \end{array}$$

In words, the winning bidder selects values for the vector of decision variables x *so as to optimize his own objective function subject to the constraint that the selected values will give the auctioneer at least the utility obtained through* b_s. *If, for some reason (e.g., discrete variables), bidder* m *can not offer* $g'(b_s)$ *the program will lead him to offer a smaller value – possibly even* $g'(b_m)$! *The auctioneer will have no problem in accepting such an offer, which goes beyond what is generally accepted out of a second-price auction.*

The auctioneer then verifies the deal *d*. In case of disagreement, the winner and the auctioneer need to further negotiate on this deal in a one-to-one fashion or offline.

Remarks:

1) In the description above the bidder supplied an offer consisting of actual values to the attributes (such as price, date, quantity etc.). In fact, all the auctioneer needs is the value of the bid in terms of his (i.e., the bidder) value function. So, it suffices that bidders supply the value of *g()* on their offers rather than the offer itself (recall that the bidders know *g()*).

2) In general the auctioneer has an incentive to reveal his true *g()* as this will encourage the bidders to make the best bids from his point of view. However, there might be situations in which an auctioneer will prefer to reveal a modified, rather than true, value function. For example, this may happen when he is engaged in 1-N negotiations at present but plans to move on to 1-1 negotiations with the same bidders in the future. If he reveals his true value function now it will make his future "opponents" knowledgeable and that might give them an advantage.

3) If the auctioneer's goal is to achieve an *efficient* outcome (that is to select the bidder with the smallest *g()* value) then his reservation price must be set to infinity. A finite reservation price is to ensure maximization of auctioneer's utility.

4) Another option which the system provides is for an auctioneer to reveal partial or modified information about his value function, let the auction procedure work as explained above but have the option for the auctioneer to select whichever bid he likes (not necessarily the one who "won" the bid in the regular sense). This is equivalent to existing bids in which the agent who issued the deal states that he is not obliged to select the cheapest

offer. In our terminology this means that he revealed part of his value function (money) but not all of it (other components may involve data on the reliability of the bidders, their tendency to meet lead times, etc.).

5) The system allows bidders to operate manually. In this case their bids are evaluated by the auctioneer and in case of winning a bidder needs provide an offer that is at least as good as the level of achievement specified as the winning values in the auctioneer's value function. Observe that the winning bidder can always simply provide the bid he presented (which is as good or better from the auctioneer's point of view).

6) In case the auctioneer sells n identical items, the best n bidders are the winners. If there are ties at the winners "low end", they are broken randomly. The highest bidder, whose bid is not equal to that of a winner, defines the deal value (if there's none, an auction-specific default rule applies).

The English Auction – Price and Price-like

We start with a brief description of the English Auction. In general, in the English auction the price of the object rises and bidders indicate whether they are willing to buy the object at that price or not. A bidder that is willing to buy at the current price is said to be an *active* bidder. At a price of *0* all bidders are active and as the price rises bidders can choose to drop out of the auction. The decision to drop out is both *public* and *irrevocable*. Thus, if bidder *j* drops out at price *p*, the bidders commonly know both the identity of the bidder dropping out and the price *p* at which he dropped out. Furthermore, once bidder *j* drops out he cannot then "*re-enter*" the auction at a higher price.

It may be useful to think of each bidder as pressing a button to signify that he is active. A number light that is publicly observed indicates the number of active participants to all bidders. Once a bidder drops out, his light goes off and the bidder cannot reenter the auction.

To allow agents to bid in such an auction, where participants are not necessarily known in advance, these ideas can be implemented with a slight variation. At any time during the auction, the active bidders can see the number of other bidders still active. In this form, the auction specified above is sometimes referred to as the "button auction." Note that while in the classical English auction the participants know both the number and the identity of the active buyers, here they know only the number but not the identity.

Remarks:

1) By making the drop out price a function of who dropped before and when, bidder takes into account the information other bidders have, as revealed by their dropping out price.

2) When values are private (see above), this auction is exactly like the second price auction and it is enough for a bidder to specify a price to drop out, as there is nothing to gain by using the information on other bidders dropping out prices.

3) It follows that the main difference between this case and the one for private value is in the strategies of the players. A drop out price for a bidder is going to be a function of the history of the play.

Strategies:

Seller: A strategy for the seller is to choose the value for r. Additionally, a seller need to specify the increments (percentage or fixed) by which the auction proceeds from round to round.

Buyer: A strategy for a buyer is to choose a drop out value as a function of who dropped out in the past and at what value.

Multi-dimensional English Auction

When agents have private values, the second price auction and the English auction, described subsequently, are equivalent from a strategic point of view. Yet it is often claimed that the English auction is more intuitive for the participants to understand and follow. While this might be true in real life, it is not necessarily so in automatic negotiation environments. To see why, we first describe the English auction. We will do it at once for the case of whole complex deals. We repeat the observation we made before, once deals are multi-dimensional, the word "auction" stands for reverse auction, bartering auction etc. as well as for the usual auction.

The Rules of the English Auction

Before the auction starts the auctioneer's (possibly modified) value function *g'()* is revealed to the bidders. The value of *g'()* is set to the reservation value set by the auctioneer and starts decreasing continuously (that is, becoming better for the auctioneer, as we are dealing with penalty functions). Bidders signal when they want to opt out. The auction ends and the clock stops when there is only one bidder left (the extreme case in which all bidders drop is discussed below). At that point, the bidder that "stayed" must produce an offer with the appropriate value (the value of g'*()* that was reached). The auctioneer must approve this offer (in verifying the offer, both sides may need to consult external data sources and other decision data). In case of disagreement, the winner and the seller need to further negotiate on this deal in a 1-1 fashion or offline.

To implement the English Auction procedure the auctioneer needs to define a "profile" that will be used to decrease his reservation value. These decreases can be based on constant absolute term (reflecting "neutral" or "indifferent" auctioneer), a constant relative term (leading to a convex shape that might be associated with an "aggressive" auctioneer), decreasing step sizes (leading to concave-shaped value functions that might signal "soft" approach towards the bidders), the decrease may depend on external parameters, etc. At any rate, given a new value *g'(b)* that was computed through the profile data, the auctioneer needs to solve the following program that tests whether this value is feasible for him. If it is feasible, it will be announced to the bidders. If not, the program will find the closest value *g'(x*)* that is possible. It will do it first by solving for the first level, if achieved exactly also for the second level with the first level value added as a constraint, and so on until at some level there was a deviation, in this case the calculation stops, or until the last level is reached. Below, we show the computation for the first level. The reason for doing it this way is that once a deviation occurs, the levels below are not important.

$$\begin{array}{ll} Lex\_Min & \{100 \cdot \delta_1^- + \delta_1^+\} \\ s.t. & g_1'(x) + \delta_1^- - \delta_1^+ = g_1'(b) \\ & GP_A \end{array}$$

Given a value function *g'(b)* stated by the auctioneer, a bidder has to determine if he stays in the auction. This is determined by computing a vector of decision variables x that satisfies the auctioneer's current requirement and provides the bidder with the best solution for his own value function *f(.)*. This is done through the following program.

$$\begin{array}{ll} Min & \alpha_i(x) \\ s.t. & g'(x) \leq g'(b) \\ & GP_B \\ & GP_A \end{array}$$

The bidder's profile allows him to choose one of four approaches to determine whether to stay in the auction in light of the optimal value *f(x*)* that was achieved as shown above.

1. The bidder may stay until he reaches his worst objective value (i.e., he stays as long as the program above is feasible, regardless of the objective).
2. The bidder may specify a percentage of the difference between his best and worst solutions. The value *f(x*)* will be compared to the value associated with that percentage and if it's still better for the bidder than this associated value, the bidder stays.
3. The bidder may specify a value (instead of the percentage in the previous approach) between his best and worst solutions. This value is used to compare against *f(x*)* as above.
4. The bidder may be unable to specify either a value or a percentage but is capable of providing an example from which we can deduce his implicit limit. The system will enable him to specify such a deal through a "deal generation" module.

A dominant strategy for the buyer in this auction is to opt out exactly at the value of *g'(.) that coincides with the buyer's private value* -- that thus leads to a zero payoff. It is easy to check that this is exactly the same bid a bidder would bid in the second price auction. However, a major difference between the second-price auction and the English auction is that in the latter the action of one bidder may depend on the actions of other bidders. We discuss this point in the context of the interdependent value model.

Closing the deal, multiple winners, disappearing bidders:

Let us assume that the seller attempted to sell n identical duplicates (n=1 is possible) of a certain item. Several situations may arise:

1. One iteration before the clock stopped (say, when the value for the auctioneer was □) there were still m>n active bidders. At the last iteration, the auctioneer decreased the value to □̄□ and only k<n bidders remained. In that case, the auctioneer leaves aside the k active bidders and returns to the m-k bidders that dropped with a value that is half way between his last two values (i.e., □̄□̄□). He will continue to drop the value (or return half-way upwards) until he either gets the additional n-k bidders he needs to complete his desired deal or until he can't slice the values by a half any longer (due to a restriction on step sizes in the value functions). In the first case, all the n "winning" bidders are committed to the same value (say, after two such iterations the value was □̄□□̄□). Note that although there were some bidders that were ready to commit to a better value for the auctioneer (there were k bidders who were ready to go to the value □̄□) all the n winning bidders must "pay the same price" to maintain a fair auction process.
2. One iteration before the clock stopped (say, when the value for the auctioneer was □) there were m>n active bidders. At the next (and last) iteration, when the value is dropped to □̄□, exactly n bidders remained. Then, the bidding process stops and the n "winning" bidders are committed to the □̄□ value.

Remark:

While the first price auction yields the same revenue to the auctioneer it is much more difficult to find an optimal strategy and hence is not recommended.

The Interdependent Value Model

General Observations

In the interdependent setup, agents should update their own valuation of the deals as they learn more about other agents' values. As a general principle in such a setup, it is optimal to employ a procedure in which as much of the agents' private information is revealed in due time to his rivals. This is why in this setup the different auctions are not equivalent any more. Indeed, when there is only one deal to agree upon, the best mechanism (the one that maximizes the auctioneer's revenue) is the *English* auction. Therefore, in this setup, it is important that all bidders know the history of the auction. We redefine the auction to take this into account.

Remarks:

1. By making the drop out price a function of who dropped before and when, bidder takes into account the information other bidders have, as revealed by their dropping out price.
2. When values are private (see above), this auction is exactly like the second price auction and it is enough for a bidder to specify a price to drop out, as there is nothing to gain by using the information on other bidders dropping out prices.
3. It follows that the main difference between this case and the one for private value is in the strategies of the players. A drop out price for a bidder is going to be a function of the history of the play.
4. We aim at automating bidders that do not necessarily know the other bidders, in that case the identity of those who drop is less important then how many dropped and when.

The Auction's rules (Restated)

1) Auctioneer's (possibly modified) value function $g'()$ is revealed to all bidders. The auctioneer chooses a reservation value r, which defines the worst value under which he is willing to perform the deal.

2) The value of r is revealed to bidders.

3) Bidders choose whether to participate or not.

4) The price clock starts falling until the point where all bidders but one dropped out. Denote the winner by m and the value at which he won by p^*.

5) The traded deal, denoted by d, *then* solves ($f_i()$ is the value function of bidder i.

$f_m(d)=min[f_m(x)]$ subject to $g(p^*) \geq g(d)$

Strategies:

We implement a dependency on the number of bidders that remain active at any given step. Thus, the bidder profile enables an optional departure from the bid process, which is given as a probability function that depends on the percentage and number of bidders who are still in the "race" at the present step. Insecure bidders may thus increase their chances of leaving the auction before it reaches its conclusion if the number of remaining bidders indicates to them that the "market" has little appreciation to the current offer by the auctioneer.

We suggest using a drop out decision table as follows:

- The table rows correspond to the current "price" in the buyer's currency, that is, his value function. Note that the buyer makes decisions in terms of his currency.
- Each column corresponds to a range of original participant numbers, for example, if column one is 2-12, it means that the auction started with 2 to 12 bidders.
- Each entry in the table includes two numbers, to be understood as "the percentage of original participants that are still present currently and the probability of dropping out". For example, suppose we observe that for a certain row 2 that corresponds to seller's value 200 and a column that corresponds to 21-30 original participants, the entry numbers are {40,0.5}. This means that when the auction value reaches 200 and the number of original participants left in the auction is 40% of the original number (20-30) that started, then drop out of the game with probability of 0.5.

The One-to-One Mechanism

This section discusses the 1-1 functionality of the mechanism layer. One-to-one is a complex trading mechanism as compared to one-to-N (auctions). Part of this complication is due to issues of symmetry -- who starts, who responds, at what intervals, how to respond and when and how to terminate.

We begin by explaining the situation in a price-only negotiation. We then outline the needed changes when more general deals, containing multi-items and multi-attributes, are treated. We define the parameters that specify the user as a negotiating entity, these include:

- *User negotiation classification parameters*
- *User operational profile*

The combination of the two dictates the actual negotiation behavior. Once executing, this information is used to dictate how to respond to the other side's offers. Here, the other side's offers are essentially values for the decision variables.

The profile is constructed in one of two ways:

- Choose from a pre-determined set of profiles (see the profiles section). This choice can then be adjusted.
- Based on answers to a set of questions, a profile is composed to express the desired behavior.

User negotiation classification parameters include:

1. Negotiation type: I offer, I respond, Symmetric (and if so, I start, the other party starts, I don't care who starts, a randomly selected starter).
2. Negotiation focus: all deal parameters, part of the parameters (could be just price), multi-phases (e.g., two phases) and for each phase, which decision variables participate in each phase.
3. Relayed information to other side: My value functions, part of my value functions, other value functions (e.g., such as the g'() function that was described above).
4. A Worst offer I'll accept (an example of such an offer is used to derive the corresponding value functions values; alternatively, it can be specified as percentage off/above average/best/worst offers, these standard offers can be calculated by the utility layer).
5. A Starting offer I'll suggest to the other side (again, this is specified via an example or via offsets from standard offers).
6. Timing parameters, the most important is time to completion of all negotiations. Another time parameter is the maximum time per single negotiation process. These parameters are used to 'speed-up' moving through

the columns (rounds) in profile tables. Specifically, if the table has N columns that have not yet been used and a round takes currently t seconds and the time to completion of this negotiation session is T, the number of columns to skip is (N/(T/t) rounded to the nearest integer. The user may also specify: a maximum skip and reaction delay to the other side's offers.

User operational profile includes the following components:

- User's tables and negotiation control data. The tables basically explain how to react to the other side's moves based on the 'round number' as well as the other side's previous offer.
- User's value functions. These functions determine how the user values offers. One possibility is a multi-level specification of objectives as done in goal programming(GP). Another possibility is a value function based on the Analytic Hierarchy Process (AHP) method (see Saaty, 1990). A third possibility is a scoring function designed for the particular application domain.

Multi-level mechanism:

For a multi-level specified value function, the one-to-one negotiations may be instructed, if so desired, to proceed on a level-by-level basis, along the value functions objective functions, where negotiation on a level starts once "agreement" about the preceding level is achieved. This also means that the time factors in negotiations should be understood in the context of level. Therefore, if we consider about 4 levels as the maximum per value functions, we can allocate 9/16 to level 1, 1/4 to level 2, 1/8 to level 3 and 1/16 to level 4.

In what follows, we first describe a mechanism for one-to-one negotiations on a single parameter, say price. In so doing, we specify the data structures used to determine a negotiating party's behavior. This is then generalized to the complex settings of multi-items, multi-attributes deals. This setting is also referred to as a multi-dimensional negotiation.

Part 1: Price Only Negotiation (from a seller's viewpoint)

(A) Assume the negotiation is on price only, and consider a seller's strategy (a mirror image strategy is specified for the buyer).

In what follows, we denote by *round* a proposal by one party and a response to it, with either Yes or No by the other party. The elicitation of parameters is described in terms of filling a 'questionnaire'. This is, of course, an abstraction and other ways are possible such as calls to an application-programming interface (API).

Check applicable entries:

1)_____ Fill in the time to end all negotiations and specify the maximum time for a single negotiation session ______.

2)______ Fill in the time delay (percentage, 5% means that you add 5% to the delay of the other side or to a *constant* delay, -5% means you are quicker to respond). This time delay may also be a function of round number, in which case a table such as Table 2 below is used.

3)______ I insist on playing only if I make all offers and the other side responds only with Yes or No answers.

4)______ I insist on playing only if the other side makes all offers and I respond only with Yes or No answers.

5)______ I insist on playing only through a symmetric mechanism. That is, if in round t the seller was the one proposing and the buyer was the responder, then in round t+1, the buyer is the one proposing and the seller is the responder. In this procedure,

5.1 ___I start

5.2___The other side starts

5.3___The starter is determined randomly

5.4___I do not care who starts

6)______ I do not care in what procedure to play.

If the procedure is not symmetric, but one in which the seller makes all offers, then point (4) in the procedure above should be deleted. If instead, the procedure is the one in which the seller only reacts by {Yes, No}, then delete all points in (B) below *except for* point (B4) which defines acceptance.

(B) Assume the procedure is symmetric, that is, the parties alternate in issuing offers (see below for a modified rule for a non-symmetric procedure), then

(B1) ____ Fill in the starting high price if you are the first to propose. If in the first round you are a responder, then fill in your starting high price in the second round as a function of the first proposal (see table below).

First round proposal in the range	My second round starting proposal
0-200	1900
200-400	1400
...	...
....	....
1200-1300	Y
	Y

In the table above one needs to specify the range of first price offers he's willing to accept. That is, if in some of the entries you specify Y instead of the number, this implies that you want to accept and hence do not specify a counter offer.

(B2) ____ Fill in the percentage decrease in each round (see different possible rules below).

(B3) ____ Fill in a floor price.

(B4) ____ Fill in the acceptance rule (assume the buyer makes the first proposal)

Round										

Accept if current proposal is equal to or greater than	00	90	60							50

In the above example, the seller will accept an offer of 360 or more in round five.

(B5) ____ Fill in the stopping rule. This rule introduces a potential cost in rejecting an offer. Introducing a probability of termination after rejection does this. In that event, the session may be over and it's unclear when and if a new session will be possible. For example, if the floor price is $100 the following table describes the probability of termination given the current rejected offer by the buyer. Filling these entries may affect both the duration and the outcome of negotiations.

Current offer	300-280	280-250	.	...	...	...	...	100
Probability of termination	.05	.10	15	.20				1.00

For example, if the price is $275 then the probability of stopping the negotiation is 0.10.

We may want to have different tables for different rounds. In particular, for a given offer p, we may want to terminate in probability q if this is round 3, but probability q' if this is round 8 (say).

Some alternatives for (B2):

(B2') Let t be the current round, let P_t denote the offer made by the buyer in round t. Finally let x_t=([P_t /P_{t-2}]-1). Fill in the following table for the percentage decrease in round t. Note that you can express some "good will" by reacting positively to the buyer. That is, if he increases his rate of raising his offers, you in return increase the rate in which you decrease your offers. Alternatively, you can interpret his "good will" as a sign of weakness and be more stubborn. Consider the following example table as filled by the seller (assume the buyer is the first to propose):

x_t\t	4	6	8						
-05	.005	.006	.05	.	...				

10	002	009	10		..				
15	00	00	15		..				
20	00	00	20		..				
	..	..	..	..					

In this table, for example, if in round 3 (say) the buyer’s price is 1.10 times his price in round one, that is $x_t=0.10$, then the seller’s price in round six will be 0.991 times his price in round two.

We may want to allow a few tables like the one above, each one for a different range of the starting proposal of the other side. For example, one table (as above) if the buyer first proposal p was such that $1000 \leq p < 2000$, and a different table is $2000 \leq p < 4000$.

(B2”) The same as in (B2’), except that a different table is constructed for different buyer’s price ranges. For example, if the current buyer’s price is up to 500, use table one, otherwise use table two.

(B2’’’) Same as (B2') but now $x_t=a*P_t/P_{t-1}+b*P_{t-1}/P_{t-2}+c*P_{t-2}/P_{t-3}$ etc.

Choose the appropriate values for a,b,c, etc.

(B2””) Let x_t be the amount (in $US, say) change (toward agreement) of the buyer in the previous round. Then the seller’s amount change toward agreement in round t+1 is $ax_t + b$. Fill in a value for *a*, and for *b*. It is also possible to allow for different *a* and *b* for different rounds. In that case there is a need to fill a table for the different a_t and b_t.

Part 2: Negotiation On Complex Deals

We start by setting the ground rules and accompany them with an illustrative example. One can envision a sequence of questions specifying the phases of negotiation and what decision variables are to be negotiated in each phase. We refer to the two negotiating parties as Bob and Sue. The presentation is from the Bob's viewpoint. Sue designates a generic opposing party. Here are the parameters Bob needs to specify.

1) Check the applicable entries:

1. ______ I'd like to negotiate the whole deal in a single phase.

2. ______ I'd like to work in a multi-phase mode (we show an example of two phases)

   i. In phase 1, I'd like to negotiate on the following attribute(s) ____ ____ using the following method _________ (can be "price only mode" or "knowledge mode" as defined below).

   ii. In phase two, I'd like to negotiate on the following attribute(s) _________ using the following method _________ (can be "price only mode" or "knowledge mode" as defined below).

As an example, suppose in phase (a) the parties negotiate on all attributes except for price and in phase (b) just on price. In phase (a), the negotiation procedure completely ignores the price attribute and therefore cannot consider price-related trade-offs as well. Except for that, negotiations are carried out in any of the "knowledge modes" detailed below. Once the deal is agreed upon, except for price, the price is negotiated upon as per part 1. Observe that this necessitates filling tables twice, once for the first "non-price" phase, and then for the second "price-only" phase. This is because in both phases there is an underlying value function.

Observe that "price only mode" necessarily treats a single attribute. In fact this attribute need not be the price attribute at all but may be any attribute that displays "price like" characteristics (i.e., it defines a "cost" to one party which equals the "benefit" to the other party).

2) a)______ I am ready to disclose my value function to the other side.

b)---------I am willing only to disclose a "subset" value function to the other side. This subset needs to be specified.

c) ---------I am willing only to disclose a "different" value function to the other side. If so, this value function needs to be specified.

d)______ I am ready to disclose my true value function only if the other side is ready to disclose his true value function.

e)___.__ I am not ready to disclose my value function.

Note that Bob needs to fill information for the true value function and for the subset to be used, or for the modified function he wishes to present to the other party.

3) In the following we examine the various possibilities of the parties' knowledge of each other's value function. We denote these functions by *f()* for Sue's function and *g()* for the Bob's function. The goal of each agent is to reach a deal with the highest possible value of his value function (in the context of GP, the highest possible value is achieved when the objective functions are minimized).

We consider four basic knowledge state possibilities:

	Bob	Sue
	K	K
	K	I
	I	K
	I	I

Here "K" means knows the other side's value function and "I" means ignorant of the other side's value function. States 2 and 3 both treat a knowledgeable party and an ignorant one and so are treated equivalently. We continue to consider the seller's problem.

Now, instead of price, or a price-like attribute, we have a value function. (When negotiations proceed level-by-level, this value function is specific to the current level that is being discussed.) In general, we have an agent, say Sue, with a value function α and another agent, say Bob, with a value function β. Observe that Sue and Bob may give different values to the same offer. We can therefore think of "Sue's currency" and "Bob's currency".

Let us review the filling up of negotiation parameters, as in (B) above, in the case of complex deals, again from Bob's point of view:

1. Negotiation type, fill the parameters as in the price only, or price-like, case.

2. Point B1, starting offer. Here Bob needs to provide an excellent offer from his point of view. The implied objective function(s) values will then be used in the actual negotiations at the appropriate level. These objective values are derived from the offer.

   Alternatively, this offer may be specified as an offset (percentage) from a standard offer (e.g. best, average) that can be generated by the utility layer by using, for example, the suggest utility.

   The situation is a bit more complex for the table detailing second offer in response to a first offer, because this is not just number filling. So, here we do not have a table specifying for each range of the other party's offers what the response would be, rather there is a standard first response no matter what the other party's offer may be. Bob's response to a first offer of another party is specified in a way similar to that of a first offer, although it need not be identical with a first offer (for example, a different percentage offset off best may be used).

3. Point B2 is the same, fill in the percentage decrease. Observe that here it relates to the overall value of an offer.

4. Point B3, floor value, is handled by asking for a worst acceptable offer (again, via an example or in terms of a standard offer). Again, the result is a specification of values to Bob's objective levels that define a worst value.

5. Point B4, acceptance rule. Here Bob needs to provide 'typical'

acceptable deals for these rounds. The system will use the objective value functions on these offers to set acceptance rules. Again, specification in terms of standard offers and offsets thereof is possible. This specification will define a level of acceptance as a function of round.

6. Point B5, stopping rule, as in the case of price only, and price-like.

7. Alternatives for (B2). Here we only treat a variant of (B2"). This variant will have three values in each table entry:

i. Percentage by which Bob is ready to worsen his own value function.

ii. Percentage by which Bob is ready to improve the Sue's offer.

iii. Percentage (may be negative), indicating response time relative to the other party's response time or a *constant time period*. The specification here should be in units that clearly indicate a strategic delay rather than a system delay; further delay should be measured above a reasonable threshold, say 2 minutes or another reasonable application specific constant. If this entry is 0, it means answer immediately.

The precise interpretation of these percentages may change depending on the primary negotiation procedure used at the utility level, as well as the knowledge mode used. For example, an ignorant may "amplify" the response percentage indicated at a table's entry, as he is "uncertain" about the quality of his intended improvement to the other party, of which he does not know the value function.

A critical issue is how to interpret the other party's offer (as used in item 7 above). When both parties are knowledgeable this is simple, and each party uses its own "currency". However, when receiving an offer from an ignorant party this is not so simple. The obvious solution is that the receiving party only uses its own currency.

However, since the other party is ignorant this may miss the other party's willingness to reach agreement and the negotiation may get stuck.

Looking then in terms of the other party's currency (i.e., how much he gave up in his own currency) seems reasonable. This information is available for a knowledgeable party. For an ignorant receiving party we can consider the possibility that this parameter is actually provided by the system, that is the system supplies information on how much the other side degraded his position in his currency (but without actually revealing the other party's value function).

	Bob	Sue	p	q	r
	K	K	use	ignore	N/A
	K	I	use	use	N/A
	I	K	use	N/A	ignore/use
	I	I	use	N/A	use

The conclusion is that we take into account the other party's state of knowledge in evaluating the improvement of his offer. The above table adds three columns to the previous table. Here *p*, *q* and *r* are parameters. They indicate the relative weights that should be assigned to percentage improvement of this buyer's offer as compared to his previous one. Parameter *p* refers to measuring the improvement in the Bob's currency. Parameter *q* refers to measuring degradations in the other party's currency. Parameter *r* also refers to degrading in the other party's currency, the difference here is that r is supplied by the other party (or the system) and is not computed based on knowledge of the other party's value function. Looking at degradations in terms of the buyer's currency is significant only when the buyer is ignorant. In that case the other party's "goodwill" can only be manifested by how much he degraded his position. The precise relative weights ascribed to *p*, *q* and *r* may be system parameters that can be further modified by advanced users.

For example, the vector $p=0.7$, $q=r=0.3$ indicates an evaluation of other party's offers that gives high weight to actual improvement as experienced by the seller and a somewhat lesser weight to the "other party's efforts". We note that giving low weights to the other party's efforts may lead to "spiraling down" negotiations. This means the other party's offer is considered as not good enough which may lead to a tough, i.e. low percentage improvement, counter offer by Bob, which may then trigger a tough response by the other party and so on.

The third line needs further explanation. The obvious choice here is "ignore" in the r column. The rational is that the other party is knowledgeable and therefore his offer should be judged in the Bob's currency only. The option of "use" will usually assign a very small weight to the other party's degradations.

Part 3: Rules for Starting the Negotiation Process

Parts 1 and 2 are described in terms of value functions. Here we treat the special important case where value functions are specified via goal programs (GPs). In this part, we assume that the negotiations are carried out by the "agents" specified by using the ignorant and aggressive utilities, as described in the section about utilities. We first explain how to compute a starting offer and then proceed to explain how to initialize three variables that are important in evaluating the other party's response (as in item 7 above).

Negotiations commence by the party who wanted to start (or at random if both parties were willing to start). The initial values in the first offer to the other party are specified as follows:

Knowledgeable Mode

- Solve: Best for me in my first objective level $(Min\,\alpha_{L_1} = \underline{\alpha}_1^*)$ s.t. my own GP
- Solve: Worst for the other party at his first objective level $(Max\,\beta_{L_1} = \overline{\beta}_1^*)$ s.t. his GP and $\alpha_{L_1} = \underline{\alpha}_1^*$
- Solve: Best for me in my second objective level $(Min\,\alpha_{L_2} = \underline{\alpha}_2^*)$ s.t. my own GP and $\alpha_{L_1} = \underline{\alpha}_1^*, \beta_{L_1} = \overline{\beta}_1^*$
- Solve: Worst for the other party at his second objective level $(Max\,\beta_{L_2} = \overline{\beta}_2^*)$ s.t. his GP and $\alpha_{L_1} = \underline{\alpha}_1^*, \beta_{L_1} = \overline{\beta}_1^*, \alpha_{L_2} = \underline{\alpha}_2^*$
- Continue this sequence of optimizations until you exhaust all levels of the objective function for both parties.

Ignorant Mode

- Solve: Best for me in my first objective level $(Min\,\alpha_{L_1} = \underline{\alpha}_1^*)$ s.t. my own GP. Notice that here (and elsewhere in places where we solve LP problems) variables that do not affect the objective function receive values that are arbitrarily assigned to them by the solver (typically at one of the end-points of their feasible ranges).
- Solve: Best for me in my second objective level $(Min\,\alpha_{L_2} = \underline{\alpha}_2^*)$ s.t. my own GP and $\alpha_{L_1} = \underline{\alpha}_1^*$
- Continue this sequence of optimizations until you exhaust all the levels of your own objective function.

Initial set-up for 1-1 negotiation

The first offer is essential for starting the negotiation. Additional values, that are necessary for the initial rounds, are chosen, for every trader as follows:

Variable name	*Trader's point of view*
Trader's last	Other party's last first offer
Other party's last	Trader's first offer
Other party's (last –1)	Other party's last first offer

These initial values enable the use of "stay-close" algorithm (see the utility section) from the very first round of negotiation. Unlike the values that are exchanged during the negotiation (whose role is to reflect "recent history" of the negotiation), here there is no history to recall and therefore these values are chosen in order to provide a reasonable starting point.

Negotiations can be carried out in a distributed way (i.e., both parties at distinct locations, exchanging messages) or a centralized way in which a "system" handles the parties' representation. In the distributed way, the trader which is the starter may not have direct access to the other party's first offer and may need to request it even though he is the real starter. In a centralized scenario the "system" has access to both and can easily perform the initialization.

Part 4: Rules for Ending the Negotiation Process

Negotiations may be stopped either due to a successful conclusion (a deal is reached), or due to technical reasons such as the passage of time, the 'end of the profile' or the lack of progress. In this part we outline a representative collection of criteria for either accepting an offer or for stopping the exchange of offers and counter offers. In case of stopping we have the option of considering the negotiations as failed or, alternatively, we may generate a compromise offer based on the current state of the negotiations as well as the original intentions of the parties.

Acceptance Criteria for Proposals:

1. Decision variables - the offer that I am going to suggest is similar to the last offer that I received from the other party.

2. Better offer – the last offer that I have received is better for me as compared to the offer I am going to suggest (according to the objective values).

3. Acceptance rules – the quality of the offer that I received is higher than the quality of offers I am ready to accept at this point in the negotiation.

4. Optimal objective – the current objective value (according to counter offer) and the optimal objective are almost equal.

5. Additional application specific rules.

Stopping rules:

1. Non-improve increase – the previous K offers I have received are pretty similar to each other, and my last K objective function values are almost identical.

2. Time limit – I have reached the deadline I specified for this negotiation process or for the intention (deal) as a whole.

3. User directive, either human generated or tool generated.

Remarks:

- When the parties end the negotiation according to the above stopping rules, it means that they didn't reach agreement. At this point, the system can provide a mediated offer that will be presented to both parties as the result of the negotiation process, along with an indication that this is a mediated offer. Two ways of constructing such a mediated proposal are presented in the utility section.

- By *similar* we mean that the offers are sufficiently close, the closeness is a system parameter, e.g., 0.75% or less difference per coordinate value, or average difference over all coordinates; other measures are possible.

Profiles in depth

A profile is composed of the following components:

1. Counter offers generation table(s). The rows of the table correspond to the percentage improvement by the other party (see the discussion above on measuring improvement) and the columns correspond to rounds. In the sequel, we describe the case of a single such table. In fact, there may be a number of tables each specifying a stage of the negotiation. A typical case may be a three-stage negotiation with three separately specified tables. In this case, acceptance in table 1 would mean starting at the first column of table two, and so on.

The table contains four parameters in each entry.

i) *Worsen mine* – the percentage by which I am willing to worsen an offer in terms of my objective functions values.

ii) *Improve opposite* – the percentage by which I am willing to improve the other side's objective functions values.

iii) *Delay time* – a positive factor indicating by how much I want to increase, or decrease, the delay in producing an offer as compared to the time it took the counter offer to reach me. Values smaller than one indicate reduction in delay time (i.e., one implies one "delay unit") while values larger than one indicate increase in delay time (i.e., 2.7 implies 2.7 "delay units").

In addition, there is a delay interval, measured in minutes, by which the delay reaction factor is multiplied; it is either 'system delay', the delay of the other party in previous move(s) and possibly averaged, or a specific unit of time, e.g., 10 minutes. The particular delay interval used may change from one table entry to another.

iv) *Probability for quitting* – the probability of quitting the negotiation at the next round. Intuitively, this is deterrence against prolonged negotiations.

All these parameters depend on the number of rounds and the percentage by which the other side improved his latest offer (see the discussion on measuring improvement above) as compared to its previous offer[55].

2. Acceptance rules. These rules specify the quality of offers a party is ready to accept, *as a function of the round number and the latest improvement* presented by the other party. When a party is ready to accept offers of a certain quality, which is specified in terms of that party's objectives (either via an example or as a percentage off/above standard offers such as optimal, average, worst).

Any offer presented to me thus far is eligible for being accepted; for example at round 6 I may decide to accept an offer made in round 2.

3. My negotiation parameters. A collection of technical parameters including, among others: (i) whether I like to disclose my value functions (in whole or in part, or a surrogate thereof), (ii) whether I want to start, and (iii) a list of parameters, (deal components) to be negotiated on.

Profile Specification Via Interrogation

So far, the description of a profile was expressed in terms of the data structure involved (a table) and the meaning of its rows, columns and table entries. Such a table may be specified in an entry-by-entry brute force, or by using a more succinct specification, which is then compiled into a full-fledged table. We present a possibility for extracting such a succinct specification.

---

[55] The parameters depend on the latest offer by the opposite side, on the cumulative percentage improvement offered by the opposite side from the beginning of the negotiation process, and over a specified "sliding window".

In order to create a profile, we present the trader with several questions.

The possible answers are (numerical scales are also possible):

(LL=very low, L=low, M=medium, H=high, HH-very high, EQ=equal):

1. As time goes on how would you like to react towards the other party?

    i. I would like to be (LL, L, M, H, HH) positive towards the other party.

    (I am under time pressure and I don't want to exert time pressure on the other party.)

    ii. I would like to be (LL, L, M, H, HH) negative towards the other party.

    (I am not under time pressure and I want to exert time pressure on the other party.)

    iii. My reactions are not influenced by the progression of time.

    (I am not under time pressure and I don't want to exert time pressure on the other party.)

2. During the rounds, would you like to:

    i. Increase (LL, L, M, H, HH) the quality of acceptable offers. (Intuitively, this means that I am not under time pressure and I want to exert time pressure on the other party.)

    ii. Decrease (LL, L, M, H, HH) the quality of the acceptable offers. (Intuitively, this means that I am under time pressure.)

    iii. The quality of acceptable offers stays the same.

    (Intuitively, this means that I am not under time pressure and I don't want to exert time pressure on the other party.)

3. If the other party improves his offer by X% (in a block as determined by the percentages range and the rounds range, see below).

   i. You are ready to improve your previous offer by a percentage, say Y% (LL, L, EQ, H, HH) X%, and all this provided that

   ii. Your new offer is not worsened, i.e., degraded, by more than a percentage, say Z% (LL, L, EQ, H, HH) X%.

4. How eager are you to close a deal (LL, L, EQ, H, HH)?

(This influences the acceptance rule and the quitting probability.)

The rational behind those questions is as follows:

- Question 1 affects how the delay specification, Worsen-mine, and Improve-opposite entries all develop as rounds progress.

- Question 2 influences on the strictness of acceptance criteria as rounds progress. The idea is that the party specifies an offer whose quality is acceptable at the outset and that the answer to this question modifies the initial behavior.

- Question 3 specifies a basic response pattern to the other party based on the other party's recent behavior. This basic response is then modified, along the rounds axis, by the answer to question 1.

- Question 4 further influences the acceptance criteria behavior along the rounds axis as well as the quitting probability.

Once a profile is specified via answers to these questions, the response is compiled into a specific table. This table is later on used to direct the party's negotiation behavior. The table compilation process details are simple and involve linear modulation along the rounds axis. The amount of modulation is specified via the modifiers LL, L, M, H, and HH. In all these examples there is a single stage and table. Rather than showing the table, we present three row segments of the table, corresponding to low, medium and high improvement by the other party.

There is still a question of fitting the profile and the knowledge modes (ignorant of knowledgeable) of the parties. One option is that the profile can be modulated by the system according to the state– knowledgeable or ignorant - and the other party's state). Another possibility is to specify four distinct profiles for the four possibilities. Restrictions are also possible, such as requiring both sides to have the same knowledge state, so as to prevent imbalances. Observe that depending on the particulars of a negotiation session it may be sometimes advantageous to be ignorant and sometimes advantageous to be knowledgeable.

In the sequel we present compilation results for three basic profiles as an example. Of course, we could construct more basic profiles. In fact, various organizations may define their very own standardized profiles. And, once a profile is defined, it can be saved and later used or modified.

Profile name: Indifferent (1)

Reference is now made to Figs. 31, 32, and 33, which are percentage tables demonstrating the indifferent profile. Observe that we do not specify any delay or quitting probabilities here. This means we produce counter offers as soon as possible and we do not impose a "termination threat". Also observe that each of the three sub-tables is specialized to a particular improvement move of the other side. The first sub-table refers to minor improvements by the other side, the second sub-table corresponds to medium improvements and the last one is associated with major improvements.

Acceptance rules describe what quality of offers is the trader ready to accept, as a function of the round number. Note that the higher the quality the better the offer. The oval in the figure below represents a starting quality level. This level may be specified by (1) presenting an example whose quality is extracted, (2) by specifying quality relative to a standard offer such as best, average, worst (the percentage may be off or above).

Quality

Reference is here made to Fig. 34 which is a simplified diagram showing the relationship between quality and number of rounds.

Profile description:

1. As time goes on how would you like to react towards the other party?

    a. I would like to be (LL, L, M, H, HH) positive towards the other party.

    (I am under time pressure and I don't want to exert time pressure on the other party.)

    b. I would like to be (LL, L, M, H, HH) negative towards the other party.

    (I am not under time pressure and I want to exert time pressure on the other party.)

    iii. My reactions are not influenced by the progression of time.

(I am not under time pressure and I don't want to exert time pressure on the other party.)

2. During the rounds, would you like to:

    a. Increase (LL, L, M, H, HH) the quality of acceptable offers. (Intuitively, I am not under time pressure and I want to exert time pressure on the other party.)

    b. Decrease (LL, L, M, H, HH) the quality of the acceptable offers. (I am under time pressure.)

    i. The quality of acceptable offers stays the same.

(I am not under time pressure and I don't want to exert time pressure on the other party.)

3. If the other party improves his offer by X% (in a block as determined by the percentages range and the rounds range, see below).

    a. You are ready to improve your previous offer by a percentage, say Y% (LL, L, EQ, H, HH) X%, and all this provided that

ii. Your new offer is not worsened, i.e., degraded, by more than a percentage, say Z% (LL, L, EQ, H, HH) X%.

4. How eager are you to close a deal (LL, L, M, H, HH)?

(This influences the acceptance rule and the quitting probability.)

Discussion:

This profile describes an indifferent trader. Answers 1 and 2 mean that reactions are fairly independent of the round number. Answer 3 implies an identical type response to the other party's moves. Answers 2 and 4 imply a low quitting probability and an average acceptance rules.

Profile name: Tough (2)

Reference is herein made to Figs. 35, 36 and 37, which are percentage tables illustrating the tough profile. Observe that we do not specify any delay or quitting probabilities here. Also observe that each of the three sub-tables is specialized to a particular improvement move of the other side. The first sub-table refers to minor improvements by the other side, the second sub-table corresponds to medium improvements and the last one is associated with major improvements.

Acceptance rules

Quality

The quality function is shown in Fig. 39.

Profile description:

1. As time goes on how would you like to react towards the other party?

a. I would like to be (LL, L, M, H, HH) positive towards the other party.

(I am under time pressure and I don't want to exert time pressure on the other party.)

b. I would like to be (LL, L, M, H, HH) negative towards the other party. (I am not under time pressure and I want to exert time pressure on the other party.)

iii. My reactions are not influenced by the progression of time.

(I am not under time pressure and I don't want to exert time pressure on the other party.)

2. During the rounds, would you like to:

   i. Increase (LL, L, M, H, HH) the quality of acceptable offers. (Intuitively, I am not under time pressure and I want to exert time pressure on the other party.)

   ii. Decrease (LL, L, M, H, HH) the quality of the acceptable offers. (I am under time pressure.)

   iii. The quality of acceptable offers stays the same.

   (I am not under time pressure and I don't want to exert time pressure on the other party.)

3. If the other party improves his offer by X% (in a block as determined by the percentages range and the rounds range, see below).

   i. You are ready to improve your previous offer by a percentage, say Y% (LL, L, EQ, H, HH) X%, and all this provided that

   ii. Your new offer is not worsened, i.e., degraded, by more than a percentage, say Z% (LL, L, EQ, H, HH) X%.

4. How eager are you to close a deal (LL, L, M, H, HH)?

(This influences the acceptance rule and the quitting probability.)

Discussion:

This profile describes a tough trader. Answers 1 and 2 mean that reactions depend on the number of rounds. Answer 3 implies a less equal type response to the other party's moves. Answers 2 and 4 imply high quitting probabilities.

Profile name: Eager (3)

The percentage table for the eager profile is shown in Figs. 40, 41 and 42. Observe that we do not specify any delay or quitting probabilities here. Also observe that each of the three sub-tables is specialized to a particular improvement move of the other side. The first sub-table refers to minor improvements by the other side, the second sub-table corresponds to medium improvements and the last one is associated with major improvements.

Acceptance rules

Quality

The quality profile for the eager profile is shown in Fig. 43.

Profile description:

1. As time goes on how would you like to react towards the other party?

    a. I would like to be (LL, L, M, H, HH) positive towards the other party. (I am under time pressure and I don't want to exert time pressure on the other party.)

    b. I would like to be (LL, L, M, H, HH) negative towards the other party. (I am not under time pressure and I want to exert time pressure on the other party.)

    iii. My reactions are not influenced by the progression of time. (I am not under time pressure and I don't want to exert time pressure on the other party.)

2. During the rounds, would you like to:

    i. Increase (LL, L, M, H, HH) the quality of acceptable offers. (Intuitively, I am not under time pressure and I want to exert time pressure on the other party.)

    ii. Decrease (LL, L, M, H, HH) the quality of the acceptable offers. (I am under time pressure.)

    iii. The quality of acceptable offers stays the same.

(I am not under time pressure and I don't want to exert time pressure on the other party.)

3. If the other party improves his offer by X% (in a block as determined by the percentages range and the rounds range, see below).

iv. You are ready to improve your previous offer by a percentage, say Y% (LL, L, EQ, H, HH) X%, and all this provided that

v. Your new offer is not worsened, i.e., degraded, by more than a percentage, say Z% (LL, L, EQ, H, HH) X%.

4. How eager are you to close a deal (LL, L, EQ, H, HH)?

(This influences the acceptance rule and the quitting probability.)

Discussion:

This profile describes a trader who's under significant time pressure. Answers 1 and 2 mean that reactions depend on the number of rounds. Answer 3 implies a great equal type response to the other party's moves. Answers 2 and 4 imply low quitting probabilities.

Summarizing the Basic Profiles

As we can see, different answers for the questions imply different profiles.

In this section we have identified three basic profiles with many possible variations around them. Each profile affects the profile counter offer generator table and the profile acceptance rules.

The three basic profiles identified are the indifferent, the eager and the tough.

Counter offer generator table:

Indifferent – the percentages are the same for all rounds.

Eager – the percentages increase during the negotiation.

Tough – the percentages decrease during the negotiation.

Acceptance rules:

Indifferent – the quality of the acceptable offers is the same at all rounds.

Eager – the quality of the acceptable offers decreases during the negotiation.

Tough – the quality of the acceptable offers increases during the negotiation.

According to these rules, there are three clear profiles that are described in this document and in the table below. The eager and the tough profiles can be emphasized with the (LL, L, HH, H, M) parameters for each decrease or increase in the first three questions.

Questions	Ind ifferent	ough	T ager	E
1. As time goes on how would you like to react towards the other party? a. I would like to be (L, M, H) positive towards the other party. (I am under time pressure and I don't want to exert time pressure on the other party.) b. I would like to be (L, M, H) negative towards the other party. (I am not under time pressure and I want to exert time pressure on the other	C	(L)	B (L)	A

party.) c. My reactions are not influenced by the progression of time. (I am not under time pressure and I don't want to exert time pressure on the other party.)				
2. During the rounds, would you like to: i. Increase the quality of acceptable offers. (Intuitively, I am not under time pressure and I want to exert time pressure on the other party.) ii. Decrease the quality of the acceptable offers. (I am under time pressure.) iii. The quality of acceptable offers stays the same. iv. (I am not under time pressure and I don't want to exert time pressure on the other party.)	C	(L)	A (L)	B
3. If the other party improves his offer by X% (In a block as determined by the percentages range and the rounds range, see below). i. You are ready to improve your previous offer by a percentage, say	EQ		L	H

Y% (LL, L, EQ, H, HH) X%, all this provided that ii. Your new offer is not worsened, i.e., degraded, by more than a percentage, say Z% (LL, L, EQ, H, HH) X%.				
How eager are you to close a deal? (LL, L, HH, H, M)?	M		L	H

Mediation, External Effects and Pre-Screening

Mediation option At any stage during the negotiations a mediation option that will go into effect only if BOTH sides agree to it. The mediation option will employ a Unification-like procedure to generate a fair proposal that takes into account the goals of sides as well as their preferences and weights. The mediation option will be facilitated through two flags installed within each cell of the profile matrix. The first flag will be denoted as: "Propose Mediation" and the second one will be denoted as "Accept Mediation". Thus, if during the negotiations the system detects that one of the users have "activated" the Propose Mediation flag, it will check to see whether the other side have activated the flag of "Accept Mediation". Only if both of these flags are active will the system issue a mediation offer. Notice that in such a case, the negotiations will stop and both sides are obligated to accept the mediator's offer.

One of the ways in which mediation may be implemented is for the purpose of fixing the values of some decision variables on which there is a mutual agreement of both parties to go to mediation. Each user may flag variables that might be negotiated through mediation during the GUI session. The system selects all the variables that were flagged by both sides and then calls a mediation procedure that fixes values for these variables. This could be done either before the negotiation started or at any time during the negotiations. The end-result of such "partial" mediation implementation is a reduction in the number of decision variables whose values are yet to be determined.

Dynamic shells – some negotiations may be affected by market-driven events that are taking place during the negotiations. For example, in trading for oil one may want to adjust his negotiation profile according to the changes that are observed in real-time in the spot and forward markets for oil. To handle such situations we will construct a shell of logical conditions that will choose one of the basic profiles according to the values observed in real-time. It should be noted that such dynamic frameworks should be supported by the utility layer which should have the capability of treating parameters such as S&P500 index, or USD-to-Euro conversion rate, etc.

Dynamic Tables - Here profile table entries may be arithmetically manipulated (e.g., multiplied) by external variables that may change over time (e.g., inventory level). Such variables may also reflect the progress of other negotiation session undertaken by the party.

Post unification screening Sometimes simultaneous 1-1 negotiations may not be allowed. In these cases we will want to screen intentions and rank them according to the likelihood of reaching a deal. The likelihood measure will be created from two sources: differences in the assignment of decision variables to levels in the objective function and differences in their target values. Denoting by S_i the set of variables that belong to intention i, L_j^i the objective level of variable j in intention i and T_j^i its target value, we define the likelihood measure between intentions i and k as: $\sum_{j \in S_i \cap S_k} \left|T_j^i - T_j^k\right| \left(\frac{1}{1+\left|L_j^i - L_j^k\right|} \right)$ The intentions are ordered in an increasing order according to this likelihood measure, i.e. the smaller the number the higher the likelihood, as illustrated in the example below (where Intention 3 is preferred to Intention 2).

Example:

Varia	Inten	Inten	Inten

ble	tion 1	tion 2	tion 3
X_1	100 (L_1)	20 (L_1)	80 (L_1)
X_2	-	40 (L_1)	40 (L_1)
X_3	30 (L_1)	120 (L_2)	50 (L_1)
Likelihood measure for Intention 1	N/A	89.48	40

Rank Computation

An objective function can be used as a *ranking function* which expresses constraints, preferences, and trade-offs on a set of attributes (decision variables). This set of attributes can be thought of as the attributes of relations in a relational database, the columns in a spreadsheet, or the attributes (i.e., class members) in an object-oriented database. The ranking function may be either *minimizing* or *maximizing*. We explain how ranking functions may be used in the context of databases. Observe that the compilation of a ranking function may be interval based (default) or target-based.

We denote a *ranking function* as a tuple $f=<o_i, di, C, P, T>$. Here o_i is the i'th objective (level), C is a set of constraints, P is a set of weighted preferences, T is a set of weighted trade-offs and d_i is either *Min* or *Max* indicating the optimization required at level i. Without loss of generality, assume $d_i=Min$. This representation will be used later on when we introduce a sub-language for defining ranking functions. Then, f, when applied to a table or database, orders a maximal subset of the table's rows so that:

- Each row satisfies the constraints in C. The extraction of satisfying rows can be done via an SQL statement. It can also be done via scanning of rows and direct application of the constraints.

- The ranking function when applied to each row is less than or equal to to the application of the ranking function of the subsequent row.

So, the rows of the table are ordered in terms of their desirability, as indicated by the ranking function. In fact, we can form a sequence of ranking functions: f_1, ... , f_n and first order the rows according to f_1 and then, given a sequence of rows with the same f_1 value, order within these rows according to f_2, and among those agreeing on both f_1 and f_2 order according to f_3 and so on. In this case we can think of $<f_1, \ldots, f_n>$ as a generalized (lexicographic) ranking function. The end result is that rows are ordered in their order of desirability.

The technique outlined above may be applied to electronic spreadsheets, as an example of a database (set of tables). Here, columns and rows play the same role as in a relational table. Examples, shown below, display a possible interface for specifying constraints, preferences and trade-offs in *Microsoft Excel*.

Additional Functionality

1. Instead of a table of data, a ranking function may be applied to a file, a spreadsheet or the result of an SQL or OQL query. In case of a file, rows may be scanned, by using some access method (sequential, index based etc.).

2. We can extend SQL (versions 2 and 3, and future versions) with a *rank clause* as follows:

Select ....

From ....

Where ....

Group by ....

Rank by *f* /* *f* is a ranking function, either compiled separately or using a *sub-language* such as the one presented subsequently */

Here, the Select clause defines the columns (attributes) of a result table. The ranking function, conceptually, operates on this result table. In ranking such a SQL statement, the optimizer may take the ranking function into account when determining how to optimize the query. Observe that if *rank by* is used then *order by* cannot be used.

3. The SQL query above may be defined as a view. Additionally, it may be defined as a view to be maintained over time. In case many such views are defined concurrently, one may maintain only the first *k* (a parameter) rows per view. Alternatively, a different *k* value may be assigned to different views based on, say, historical usage. Observe that a user may define a number of similar views differing only in their ranking function.

4. The ranking function may be defined in terms of external variables, i.e., not in the database. For example, suppose each row in a table is associated with a column Company. Suppose further that each company is traded on a stock market. Then, one can write *stock value(Company)* and this is interpreted by an external function that checks the current stock value of the company of *this* row.

5. Comparison between two, or more, ranking functions, all applied to the same table. Graphically, deviations from ordering according to the first ranking function are indicated.

6. The ranking functions may be built *incrementally* where at each stage the new order is calculated and shown relative to the previously computed one(s).

7. Specification of default values via rules. This feature is useful especially in the case of null values. Here, rules can be applied, on a row or table basis, for the generation of such values. The generated values can replace null values or existing values, as indicated by rules.

8. Values may be specified for hypothetical columns. Here, one can add column(s) that are not present in the original table and specify values, via rules as above, for these columns.

9. Values may also be specified for rows. This is the idea of *generate and test*. Tuples (rows) are generated as needed. This may be in addition to actual physical tuples present at the table. Alternatively, the whole table content (i.e., its rows) may be non-physical (i.e., generated).

10. Aggregation-based-objective functions may be specified. Here, the conditions, and rules, are based on aggregations (e.g., AVG, MAX, MIN).

11. Ranking functions may be synthesized out of examples. Here, given a table, a subset of the rows is ordered according to their desirability (preferably, the top rows). The idea is to fit the parameters of ranking functions so that the synthesized functions derive the specified order of desirability. The resulting function may be then applied to test rows in order to verify the synthesis. The fitting of parameters may be done in many ways (e.g., neural networks, regression), one such way is as described below.

12. The trade-off compilation is penalty oriented. That is, we "punish" from being "away from the trade-off line". Alternatively, we can use a "substitution based" trade-off (still using the same syntax). For example, we can trade m units of, say, X_i for n units of, say, X_j. Mathematically, we can introduce new variables, x_is, and use them as the resulting values for the X_is variables. Observe that this provides the underlying solver with an opportunity to take actual values of the X_is variables and "think" about them in terms of the new x_i variables. Consider how we need to alter the original goal constraints by using the new variables and adding conservation equations:

$X_i + D^-_i - D^+_i = V_i$ */** Original i'th goal constraint **/*

$X_j + D^-_j - D^+_j = V_j$ */** Original j'th goal constraint **/*

$x_i + D^-_i - D^+_i = V_i$ */** Use new variable and form new goal constraint **/*

$x_j + D^-_j - D^+_j = V_j$ */** Use new variable and form new goal constraint **/*

*/** Add a "conservation equation", note that it's linear **/*

$m(x_i - X_i) = n(X_j - x_j)$ */** What we "gained" in X_i we "give" in X_j **/*

The translation depicted above can be applied to more than two variables by simply writing a more complex conservation equation for each trade-off statement. Observe that in case of a number of trade-off statements, each one is compiled separately. We have two options here that imply different semantics. The first is to use the same x_is in these separate trade-off statement compilations. The second is to use distinct new x_is in each such translation. Both semantics have their advantages and disadvantages. A major drawback of the second option is that there is no unique deviation variable for X_i as there are many new distinct x_is. The question is which deviation variable to use in the original objective function. One possibility is to use an "average deviation". However, this is not very convincing which lends more credibility to option one.

Synthesizing ranking functions

We consider the problem of synthsizing a ranking function from example preferences. The first issue to consider is obtaining the desired values V_i for attribute X_i. The simplest case is when these values are provided explicitly by the user. Next, the user may indicate that (1) desirability is high at a point (target) and decreases for botth decreasing and increasing values, (2) desirability is linear, (3) a more complex pattern, for example, desirability is constant on an interval and decreases on both sides.

Case 1: We employ the following heuristics. For each attribute X_i, the desired value, V_i, is determined as the average value for that attribute over the first (i.e., most preferable) k (a parameter, usually 2 or 3) rows. For each attribute X_i, a target equation of the form $X_i + D^-_i - D^+_i = V_i$ is introduced. Here, D^-_i, D^+_i are *deviation variables*.

Generate inequalities, representing the order of desirability of rows. Consider for example two consecutive (in terms of desirability) rows: *r* and *s*. For each deviation variable, calculate its actual deviation based on the values for columns in rows *r* and *s*. Suppose, without loss of generality that there are only two variables X_1 and X_2.

Suppose that the deviations are as follows:

Row/ Deviation for	D^-_1	D^+_1	D^-_2	D^+_2
r	b	0	0	c
s	d	0	e	0

Observe that in both rows there is "undershooting" for variable X_1, whereas for variable X_2, there is "overshooting" for row *r* and "undershooting" for row *s*. The resulting inequality is: $C_{11}\,b + C_{12}\,0 + C_{21}\,0 + C_{22}\,c < C_{11}\,d + C_{12}\,0 + C_{31}\,e + C_{33}\,0$. Similarly, generate such an inequality for each pair of consecutive rows. Further add inequalities indicating that each coefficient is bigger than or equal to zero.

We consider the set of first ranked *m* (a parameter) rows and generate inequalities for each pair of consecutive rows. Let *S* be the set of resulting inequalities. Solve *S* and obtain values for the coefficients C_{ij}'s. If some of these coefficients are not uniquely determined, choose arbitrarily within their domain of satisfaction, for example, if $(1 < C_{22} < 5)$ then choose $C_{22} = 3$. If *S* is inconsistent, one of the *m* rows is dropped and the procedure is retried. (If too many rows (a parameter) are dropped this way, the procedure *fails*.) Next, test the values obtained in solving *S* by ranking the remaining rows. If the resulting order is acceptable (as tested on other rows), a solution is achieved. Otherwise, there must be two "offenders" that are wrongly ordered. The procedure is redone with the two new "offenders" included in the group of rows used in generating inequalities. This process may either lead to an acceptable ranking function; alternatively, it may fail to produce such a function.

Introducing equations for trade-offs may enhance the procedure above by providing additional degrees of freedom. This is because the introduction of trade-offs simply manifests itself with additional components, namely trade-off deviation variables, within objective functions.

Case 2: The user indicates that the preference function is a simple linear additive function (as opposed to the more complex penalty functions described above). Given a vector of multi-attribute alternatives ("rows"), ordered by their desirability to the decision maker, we seek to determine a unique vector of weights to be associated with the attributes. We assume a linear scoring (or utility) function:

$$S_i = \sum_j a_{ij} \cdot w_j$$

Where a_{ij} are the data (value of attribute j in alternative i), S_i is the score of alternative i and w_j is the weight of attribute j.

The decision maker (DM) is capable of expressing pairwise preferences among any pair of alternatives. Such comparisons would lead to k preference constraints out of a given set of m alternatives where $k \le \frac{m \cdot (m-1)}{2}$. This inequality is expected to be strong since in some cases the DM is unable (or unwilling) to express preferences between some of the possible pairs of alternatives. We denote the set of preferences as P.

To obtain the set of weights that will result in maximal discrimination power among the alternatives we implement a Max_Min modeling approach as follows:

$$\begin{aligned}
&Max \quad \varepsilon \\
&s.t. \\
&\qquad S_i - S_h \ge \varepsilon \qquad \forall \{i,h\} \in P \\
&\qquad S_i = \sum_j a_{ij} \cdot w_j \qquad \forall i \\
&\qquad \sum_j w_j = 1 \\
&\qquad w_j \ge 0
\end{aligned}$$

A positive optimal value ($\varepsilon^* > 0$) indicates that we found a set of weights that clearly separates each pair of alternatives for which preference was expressed. A zero-value optimal solution indicates that the resultant weight vector leads to one or more weak preferences relations and a negative value for the optimal solution (notice that epsilon is unbounded) indicates inconsistency on the part of the DM in the way he described his pairwise preferences.

The formulation above might be refined if we attach an ever-decreasing level of importance to the preference constraints as we move down the ordered list. Then, the objective function can be reformulated as a weighted sum of differences rather than maximum of the minimal difference:

$$
\begin{array}{ll}
Max & 10^{k-1}\varepsilon_1 + 10^{k-2}\varepsilon_2 + \ldots + \varepsilon_k \\
s.t. & \\
 & S_1 - S_2 \geq \varepsilon_1 \\
 & S_2 - S_3 \geq \varepsilon_2 \\
 & \ldots\ldots \\
 & S_{k-1} - S_k \geq \varepsilon_k \\
 & S_i = \sum_j a_{ij} \cdot w_j \qquad\qquad \forall i \\
 & \sum_j w_j = 1 \\
 & w_j \geq 0
\end{array}
$$

Case 3: A more complex pattern. A heristics here is to assume that the top *m* (a parameter) rows delimit the interval in which desirability is high. This determines the compilation of preferences and the coeficients are determined as in case 1.

A Ranking Sub-language

This sub-language can express ranking functions in a textual form. It can then be used as an extension to SQL or to other programming or database languages, e.g., OQL.

Constraints

Consider a single database table. The constraints part can be thought of as a *filter* applied to a table of data. For example *(X=7) AND (Y>6)* where *X* and *Y* are table columns. The constraints we consider include integer/linear inequalities, simple inequalities, membership in a set and more. In fact, the precise constraint language is dependent upon the constraint solver component we utilize. There are many possibilities for such solvers, e.g., ILOG's CPLEX or ILOG's CSP solver.

Preferences

Preferences indicate where, within the constraints, are values more preferable. For example, *Prefer(X=8)* indicates that the preferred value for *X* is *8*, and *Prefer(Y, 7,9)* indicates that the interval *[7,9]* is the preferred one for values of *Y*. Preferences may be associated with ***weights***. For example, *0.7:Prefer(Y,7,9)* indicates that the interval *[7,9]* is preferred with *importance factor* 0.7. In general, an importance factor can be any non-negative number.

Deviations

A deviation is expressed within the context of a preference. A deviation is *oriented* as either $^+$ or $^-$. For example $dev(Prefer(X=8))^+$ indicates a deviation in the positive direction from *X=8*, that is, a value of *X* larger than *8*. Deviations may be associated with *weights*. For example, $0.7{:}Prefer(X=8)^+$ indicates that a deviation from the value *X=8* in the positive direction has *importance* 0.7. Deviations may also be associated with intervals as in $0.7{:}Prefer(X,7,9)^+$. A preference with no indicated deviation is assumed to have default weights associated with positive and negative deviations. One possibility is associating a weight of *0.5* with both positive and negative deviations although other default settings are possible.

Trade-offs

A *simple trade-off* is of the form *trade-off([a,b],X,[c,d],Y,(x,y), mX for nY)*. It means that when *X* is in the interval *[a,b]* and *Y* is in the interval *[c,d]*, *m* units of *X* are *tradable* for *n* units of *Y*. Further, the point *(X=x, Y=y)* with *x* in the interval *[a,b]* and *y* in the interval *[c,d]*, fixes the trade-off line. Here too, trade-offs may be weighted, for example, *0.5:trade-off([1,10],X,[100,120],Y,(110,2),3X for 2Y)* indicates, with an importance factor of *0.5*, that in the appropriate intervals and point, *3* units of *X* are tradable for *2* units of *Y*. More general trade-off statements involving more than two attributes are possible. For example, *0.5:trade-off([1,10],X, [100,120],Y,[50,400],Z,(2,101,58), X for 2Y + 3Z)* indicates that in the appropriate ranges and point, a unit of *X* is tradable for *2* units of *Y* plus 3 units of *Z*.

The penalty for deviating from the trade-off may depend on whether the deviation from the trade-off is "above" or "below". If nothing is indicated than "below" and "above" deviations are treated equally. Consider a trade-off, where *X* is Dollars and *Y* is days, of the form: 0.5: *trade-off([1,1000],X,[100,120],Y,(100,102),-100X for 1Y)* indicating readiness to pay $100 less per each extra day. Now, an actual *(Dollar amount, Delivery day)* pair chosen maybe "above", i.e., less than $100 was discounted for an extra day, or "below", i.e., more than $100 was discounted for an extra day. To differentiate between these cases, we again use the $^+$ (for above) and $^-$ (for below) superscripts. So, we may write:

0.8: *trade-off([1,1000],X,[100,120],Y,(100,102),-100X for 1Y)*$^+$ and

0.1: *trade-off([1,1000],X,[100,120],Y,(100,102),-100X for 1Y)*$^-$, indicating that not discounting enough is less desirable (higher weight) than a severe discount (alternatively, we could refrain for penalizing for a deeper discount at all).

Objective Functions

Each objective function can be specified using a general programming statement. In this presentation we only present single level objective functions.

Ranking Function Example

First, let us consider a simple example. The example describes the implementation of a *ranking function* on a single table that substantially simplifies the exposition. The simple example will be followed by a more general description.

Imagine the given *Flights* table and *C* containing the following *constraint*:

Flights.Stops<2

(A nonstop or one stop flight.)

And the following set of *Preferences P*, *Deviations* and *Trade-offs T*:

4.0:Prefer(Flights.Stops=0)$^{+}$ (I don't want intermediate stops.)

2.7:Prefer(Flights.Price=1000)$^{+}$ (I don't want to pay much more.)

0.3:Prefer(Flights.Price=1000)$^{-}$ (Too cheap means low quality.)

1.5:Trade-off([0, 2,000],Flights.Price, [0,3],Flights.Stops ,(1000,0), -100 Price for 1 Stops)$^{+}$

(For each additional stop, within the interval, I expect a $100 discount on the price.)

First let us consider the *Flights* table records satisfying the *constraint*.

Record #	Carrier	Origin	Destination	Stops	Class	Price	Departure	Arrival
1	AF	LV	LHT		Economy	$675.00	09:00 AM	12:25 PM
2	AF	LV	LHT		Business	$1,200.00	09:00 AM	12:25 PM
3	BA	LV	LHT		Economy	$725.00	08:00 AM	11:00 AM
4	BA	LV	LHT		Business	$1,200.00	08:00 AM	11:00 AM
5	Lufthansa		LHT		Economy	$550.00	06:00 AM	01:00

		LV					PM
6	Lufthansa	LV	LHT	Business	$875.00	06:00 AM	01:00 PM
7	LY	LV	LHT	Economy	$530.00	07:00 AM	10:15 AM
8	LY	LV	LHT	Business	$900.00	07:00 AM	10:15 AM
9	Swiss Air	LV	LHT	Economy	$700.00	08:10 AM	01:30 PM
10	Swiss Air	LV	LHT	Business	$1,165.00	08:10 AM	01:30 PM

Next, we rank the above resulting table based on the set of *Preferences, Deviations* and *Trade-offs* we specified, resulting in the following *Ranked table*. Notice that the order of records in the ranked table is different when compared with the original one. For example, records that represent a nonstop flight with the closest price distance below $1,000 are ranked highest, thus appear at the top.

Record

#	Carrier	Origin	Dest.n	Stops	Class	Price	Depart.	Arrival
8	LY	LV	LHT		Business	$900.00	07:00 AM	10:15 AM
3	BA	LV	LHT		Economy	$725.00	08:00 AM	11:00 AM
7	LY	LV	LHT		Economy	$530.00	07:00 AM	10:15 AM
4	BA	LV	LHT		Business	$1,200.00	08:00 AM	11:00 AM
6	Lufthansa	LV	LHT		Business	$875.00	06:00 AM	01:00 PM
9	Swiss Air	LV	LHT		Economy	$700.00	08:10 AM	01:30 PM
1	AF	LV	LHT		Economy	$675.00	09:00 AM	12:25 PM
5	Lufthansa	LV	LHT		Economy	$550.00	06:00 AM	01:00 PM
10	Swiss Air	LV	LHT		Business	$1,165.00	08:10 AM	01:30 PM
2	AF	LV	LHT		Business	$1,200.00	09:00 AM	12:25 PM

The ranking was based on the *Ranking function (f=<Min, C, P, T>)*. In this case the objective is:

$$Min\ (Z_{Stops\ Preference} + Z_{Price\ Preference} + Z_{Price\text{-}Stops\ Trade\text{-}off})$$

The objective function formula for $Z_{Stops\ Preference}$ is:

```
If $Flights.Stop>0$ Then $Z_{Stops\ Preference}=4*Flights.Stops$
Else $Z_{Stops\ Preference}=0$
```

The objective function formula for $Z_{Price\ Preference}$ is:

```
If $Flights.Price>=1000$
    Then $Z_{Price\ Preference}=2.7*(Flights.Price-1000)/1000$
        Else $Z_{Price\ Preference}=0.3*(1000-Flights.Price)/1000$
```

And the relevant portion of the objective function formula for $Z_{Price\text{-}Stops\ Trade\text{-}off}$ is:

```
If $Flights.Stop=0$ Then
    If $Flights.Price>=1000$
        Then $Z_{Price\text{-}Stop\ Trade\text{-}off}=1.5*(Flights.Price-1000)/1000$
    Else $Z_{Price\text{-}Stop\ Trade\text{-}off}=0$
Else If $Flights.Stop=1$ Then
    If $Flights.Price>=900$
        Then $Z_{Price\text{-}Stop\ Trade\text{-}off}=1.5*(Flights.Price-900)/900$
        Else $Z_{Price\text{-}Stop\ Trade\text{-}off}=0$
```

The formulas above are slightly different than the compilation described previously. Here, the deviations are measured by normalizing the absolute deviation distance relatively to the expected point on the trade-off line (hence the division once into 1000 and once into 900). The table below depicts this computation:

Record #	Price-Stops Trade-off	Stops Preference	Price Preference	Z (Sum of z)
8	0	0	0.03000	0.03000
3	0	0	0.08250	0.08250
7	0	0	0.14100	0.14100
4	0.30000	0	0.54000	0.84000
6	0	4	0.03750	4.03750
9	0	4	0.09000	4.09000
1	0	4	0.09750	4.09750
5	0	4	0.13500	4.13500
10	0.44167	4	0.44550	4.88717
2	0.50000	4	0.54000	5.04000

A more complex example may consider two or more tables with a more complex set of *constraints, preferences, deviations* and *trade-offs*. In this case, the method of calculating the resulting table, will start with enforcing the *constraints*, creating a *join table* consisting of the constraints-satisfying data and ranking the resulting *join table* based on the set of *preferences, deviations* and *trade-offs*.

Deployment Example, Excel

Below, we provide a deployment example within the *Microsoft Excel* environment. The example describes a possible *Excel* environment GUI for the definition of *constraints*, both continuous and discrete, *preferences, deviations* and *trade-offs*.

A typical flights table is shown in Fig. 44. A typical hotels table is shown in Fig. 45. A typical car rentals table is shown in Fig. 46.

Constraints, Preference and Deviations (continuous attribute)

Reference is now made to Fig. 47, which is a simplified diagram showing a constraints and preferences form in front of the flights table, and illustrating how it can be used to select preferred flights. In the form, we have the following components:

Ranking function (f=<d, C, P, T>): *Vacation (d=Min)*

The table of interest in this case: *Flights*

Constraint expression:

Flights.Stops<1 (A nonstop flight.)

Preferences expression:

$4{:}Prefer(Flights.Stops=0)^{+}$

$0{:}Prefer(Flights.Stops=0)^{-}$

Note that in section 1 we had a single *Importance* factor, in this form it is broken into an overall *Importance* (in this case *4*) and the separate specification of both *Positive* and *Negative Deviations* factors. In this case the importance of deviation for each stop in the positive direction, above zero, is 1 while the importance of deviation of each stop in the negative direction, below zero, is 0 (since the number of stops cannot be less than 0). Thus, in this case, the overall importance for *Deviation* $^{+}$ is 1*4=4).

The result for the *Flights.Stops<1 constraint* includes the following records:

Carrier	Origin Dest.	Stops	Class	Price	Departure	Arrival
BA	LVLHT		Economy	$725.00	08:00 AM	11:00 AM
BA	LVLHT		Business	$1,200.00	08:00 AM	11:00 AM
LY	LVLHT		Economy	$530.00	07:00 AM	10:15 AM
LY	LVLHT		Business	$900.00	07:00 AM	10:15 AM

Since all the records satisfying the *constraint* are with *Flights.Stops=0,* they all get the same ranking.

Constraints, Preference and Deviations, (discrete attribute)

Reference is now made to Fig. 48, in which a form is shown for selecting a car hire offer in accordance with user defined preferences. In the form, we have the following additional components:

Ranking function (f=<d, C, P, T>): *Vacation (d=Min)*

The table of interest in this case: *CarRental*

Constraint expression:

CarRental.Pickup=('Airport', 'Train Station', 'Hotel')

Preferences expression:

1.5:Prefer(CarRental.Pickup='Hotel')

Preferences expression:

2.4:Prefer(CarRental.Pickup='Train

Station')

Again, the single *Importance* factor is broken into an overall *Importance* (in this case 3.0) and the separate specification of a *fraction* associated with each item. The larger the *fraction*, the unhappier you are with that option. This *fraction* is then multiplied by *Importance* to generate an overall *Importance*.

The result for the *CarRental.PickUp=('Airport' or 'Hotel' or 'Train Station') constraint* includes the following records:

Company	Pick Up	Return	Class	Price
Avis	Airport	Airport	D	$42.00
Avis	Hotel	Hotel	D	$60.00
Budget	Airport	Airport	D	$18.00
Hertz	Airport	Airport	D	$17.00
Hertz	Hotel	Hotel	D	$55.00
One	Airport	Airport	D	$22.00
Sixth	Airport	Airport	D	$19.00

Since the *CarRental.PickUp='Airport'* is preferred over the *CarRental.PickUp ='Hotel'* we get the following ranking:

Company	Pick Up	Return	Class	Price
Avis	Airport	Airport	D	$42.00
Budget	Airport	Airport	D	$18.00
Hertz	Airport	Airport	D	$17.00
One	Airport	Airport	D	$22.00
Sixth	Airport	Airport	D	$19.00
Avis	Hotel	Hotel	D	$60.00
Hertz	Hotel	Hotel	D	$55.00

Trade-offs

Reference is now made to Fig. 49, which is a simplified diagram showing a form for setting preferences for ranking flights according to preference.

In the form of Fig. 48, we have the following components:

Ranking function (f=<d, C, P, T>): *Vacation (d=Min)*

The table of interest in this case: *Flights*

The attributes of interest in this case: *Stops, Price*

Trade-off expression for *Deviation* +:

2.4:Trade-off([0, 2,000], Flights.Price, [0,3],Flights.Stops,(1000,0), -100 Price for 1 Stops) $^+$

Trade-off expression for *Deviation* -:

0.6:Trade-off([0, 2,000], Flights.Price, [0,3],Flights.Stops,(1000,0), -100 Price for 1 Stops) $^-$

The expression indicates that the importance factor for positive deviation is *2.4 (3*0.8)* while the importance factor for the negative deviation is *0.6 (3*0.2)*, in the appropriate intervals; each additional stop discounts $100 of the flight price.

Again, the single *Importance* factor is broken into an overall *Importance* (in this case 3.0) and the separate specification of the *Positive* and *Negative Deviations* factors.

1. Problem Statement

We consider the following optimization problem. We are given a set of m suppliers. Initially, we assume that each supplier, S_i, can sell items of a *single* type; the cost of a unit of the item depends on the number of units ordered from S_i. Suppose that S_i can sell items with m_i different costs per unit. These costs are given in the *price table* of S_i, in which the l-th entry gives the cost of a unit of the item in range l, $1 \leq l \leq m_i$. The l-*th* entry in the table contains an interval

$[r_{i,l}, u_{i,l}]$ and the cost per unit, denoted by $c_{i,l}$. This cost will be used when the number of units supplied by S_i is $r_{i,l} \leq x \leq u_{i,l}$. The maximal number of items which S_i can supply is given by T_i, $1 \leq i \leq m$. Suppose that we need to supply n units of the item to some client. The *deal splitting (DS)* problem is to determine how many units will be supplied by S_i, $1 \leq i \leq m$, such that the *overall* cost of the deal is *minimized*, and the total number of supplied items is *at least* n.

Formally, let n be the size of the order and let $cost_i(x_i)$ be the total cost of x_i units of the item when supplied by S_i; then, we need to find a set of integers $x_1, \ldots, x_m$ such that $\sum_{i=1}^{m} x_i \geq n$ and $\sum_{i=1}^{m} cost_i(x_i)$ is minimized, where $cost_i(x_i)$ is the cost

of buying x_i units from S_i.

We use in our study two common properties of price tables.

Definition 1.1 (Monotonicity): The price table of S_i is *monotone* if for any $1 \leq l \leq m_i - 1$

$$c_{i,l+1} \leq c_{i,l}.$$

The monotonicity condition captures the tendency of the market to favor larger orders: thus, the cost per unit cannot increase if we buy more units of the item.

Definition 1.2 (Rationality): The price table of S_i is *rational* if for any $n_1, n_2 \geq 1$,

$$n_1 < n_2 \Rightarrow cost_i(n_1) \leq cost_i(n_2) .$$

In words, the rationality condition implies that if we need n_1 units of the item, it never pays to buy $n_2 > n_1$ units from S_i. A common example of a rational table is the *"Buy one, get one free"* rule, in which we pay the *same* total price for one or two units of the item. Note that when buying two units we get that the price per unit is half the original price; however, since our measure is the total cost of the deal, if the order size is $n=1$, we would buy a *single* unit.

Remark: Note that when we cannot buy s units of the items from the supplier S_i, for some $1 \leq s \leq T_i$, we define $cost_i(s) = \infty$. Therefore, if the price table of S_i is monotone and rational, then we implicitly assume that it is also continuous in the range $[1, T_i]$. We can buy from S_i any number of units in this range.

2. Hardness Result

The DS problem is NP-complete. This holds also in the special case where the price tables satisfy the monotonicity and rationality conditions, as given in Definitions 1.1 and 1.2.

The following is a sketch of the reduction from PARTITION [GJ-79] used in the proof of hardness:

Input: A set of elements, A, where the element a_i has the size $s(a_i)$, and $\sum_{i \in A} S(a_i) = B$

Question: Is there a subset of the elements $A' \subseteq A$, such that $\sum_{i \in A'} S(a_i) = B/2$?

For the above instance of PARTITION we define an instance for DS. We need to supply at least $n = B/2$ units from the item. There are $|A|$ suppliers: S_i can supply at most $s(a_i)$ units from the item; if S_i sells less than $s(a_i)$ units, the cost per unit is $1 + 1/(s(a_i)-1)$, otherwise the cost per unit is 1.

Claim: *There is a partition iff there is a deal with the total cost B/2.*

3. Optimal Algorithms for Restricted Deal Splitting

3.1 A Single Item

In the *restricted deal splitting (RDS)* problem we need to solve the DS problem, with the additional constraint that the number of suppliers participating in the deal is at most $1 \le k \le m$, where k is a fixed constant. The following is a polynomial time algorithm that solves optimally the RDS problem.

We define for each supplier, S_i, a set of m_i sub-suppliers: the cost per unit of sub-supplier $S_{i,l}$ corresponds to the l-th entry in the price table of S_i.

Algorithm IP proceeds in two stages:

1. Guess a subset S_p of at most k sub-suppliers that will participate in the deal.
2. Find an optimal deal for the subset S_p.

We now describe in further detail stage 2. For convenience we represent our problem as an *integer program (IP)*. Let $r_{i,l}$ be the minimal number of units of the item that can be supplied by $S_{i,l}$; $u_{i,l}$ is the maximal number of units that $S_{i,l}$ is allowed to sell. Denote by T_i the maximal number of units of the item that can be supplied by S_i; then, w.l.o.g. we assume that $T_i = u_{i,m_i}$.

The cost of a unit of the item when supplied by $S_{i,l}$ is denoted by $c_{i,l}$. We assume in our discussion of the single item case, that the price tables are monotone. This implies that in any optimal solution, each supplier would be represented by at most one sub-supplier. Finally, the number of units sold by $S_{i,l}$ is denoted by $x_{i,l}$, where $x_{i,l} \geq 0$ is an integer.

In stage 2. we need to solve the following IP:

$$\text{Minimize} \quad \sum_{i=1}^{m} \sum_{l=1}^{mi} c_{i,l} x_{i,l}$$

$$\text{Subject to} \quad \sum_{i=1}^{m} \sum_{l=1}^{mi} x_{i,l} \geq n$$

$$x_{i,l} \geq r_{i,l} \quad S_{i,l} \in S_p$$

$$x_{i,l} \leq u_{i,l} \quad S_{i,l} \in S_p$$

$$x_{i,l} = 0 \quad S_{i,l} \notin S_p$$

The above program is solved optimally by the following greedy procedure:

(i) For any sub-supplier $S_{i,l} \in S_p$, assign to $S_{i,l}$ $r_{i,l}$ units; update $u_{i,l} = u_{i,l} - r_{i,l}$. If the number of assigned units is at least n, stop.

(ii) Sort the sub-suppliers in S_p in non-decreasing order, by cost per unit.

(iii)Starting from the first sub-supplier in the list, assign to each sub-supplier the maximal possible number of units, until the overall number of units is at least n.

It can be shown (by interchange argument) that the above procedure yields a deal whose cost is minimal.

The complexity of the algorithm IP is determined by the number of guesses plus sorting the sub-suppliers (which can be done once, as a preprocessing step).

Since the price tables are monotone, we can get a sorted list of all the sub-suppliers by merging m sorted lists, where the i-th sorted list represents the m_i sub-suppliers of S_i. This can be done in $O(\log m \cdot \sum_{i=1}^{m} m_i)$ steps. Let $m_s = max_i m_i$ be the maximal number of sub-suppliers of any supplier. Then the overall running time of the algorithm is

$$O(\log m \cdot \sum_{i=1}^{m} m_i + \sum_{i=1}^{m} (m \cdot m_s)^i) = O(\log m \cdot \sum_{i=1}^{m} m_i + (m \cdot m_s)^{k+1}).$$

3.2 Multiple Items

When the deal consists of R item types, for some $R \geq 1$, we define a sub-supplier for each possible combination of the items, with a corresponding range (i.e., number of units from each item), such that the cost of a unit of each item is fixed.

We assume that each supplier, S_i, $1 \leq i \leq m$, can provide at most $T_{i,j}$ units from item j, $1 \leq j \leq R$.

In this generalized version of the problem we define monotonicity and rationality as follows.

Definition 3.1 (Monotonicity-M): The multi-item price table of S_i is *monotone* if, for all $1 \leq j \leq R$, the price per unit cannot increase if we increase the number of units of item j bought from S_i. Formally, for any $1 \leq l_1, l_2 \leq m_i$, if we buy n_1, n_2 units of item j from sub-suppliers l_1, l_2 respectively, then

$$n_1 < n_2 \implies c_{i,l_1,j} \geq c_{i,l_2,j}$$

for all $1 \leq j \leq R$, where $c_{i,l,j}$ denotes the cost of buying a unit of item j from the

l-th sub-supplier of supplier *i*.

For defining the rationality condition we denote by the *R-tuple* $(a_1, \ldots, a_R)$

a combination of the R items supplied by S_i, that is, a_j units are supplied from item j. Denote by $cost_i\,(a_1, \ldots, a_R)$ the total cost of the *R-tuple* $(a_1, \ldots, a_R)$ *when* supplied by S_i; $cost_i\,(\cdot)$ applies to one of the ranges of S_i.

Definition 3.2 (Rationality-M): The multi-item price table of S_i is *rational* if for any pair of *R-tuples* $(a_1, \dots, a_R)$, $(b_1, \dots, b_R)$, if $(a_1, \dots, a_R) < (b_1, \dots, b_R)$ (coordinate-wise), then

$$cost_i\,(a_1, \dots, a_R) \leq cost_i\,(b_1, \dots, b_R).$$

Note that, in general, we do not require any of the above properties (i.e., monotonicity or rationality) to hold.

Example 3.2: Suppose that $R=3$ and the items are *printers* (item no. 1), *cartridges* (item no. 2) and *paper boxes* (item no. 3). The following is the price table of the supplier S_1.

	printers	cartridges	paper
range	[0,2]	[0,5]	[0,9]
unit cost	300	30	15
range	[2,5]	[7,9]	[10,100]
unit cost	280	25	10
range	[6,20]	[10,50]	[10,100]
unit cost	250	23	10

Table 3.2: A price table for multiple (3) items.

Note that we do not require *continuity* in the number of units that we can buy from an item, when moving from one sub-supplier to another. Thus, for example, in Table 3.2, the sub-supplier $S_{1,1}$ can supply upto *5* units of item 2, while $S_{1,2}$ supplies at least *7* unit. (Therefore we cannot buy *6* units from S_1).

For the IP formulation we define for S_1 *three* sub-suppliers: $S_{1,1}$, $S_{1,2}$ and $S_{1,3}$. For $S_{1,1}$ each line below presents the cost per unit and minimal/maximal number of units sold from each item (e.g., the first line refers to item *1*, i.e., *printers*).

$$c_{1,1,1} = 300; \; r_{1,1,1} = 0; \; u_{1,1,1} = 2;$$

$$c_{1,1,2} = 30; \; r_{1,1,2} = 0; \; u_{1,1,2} = 5;$$

$$c_{1,1,3} = 15; \; r_{1,1,3} = 0; \; u_{1,1,3} = 9;$$

Similarly, we define prices and ranges for $S_{1,2}$ and $S_{1,3}$. Suppose that S_1 can provide overall 40 printers, 100 cartridges and 300 paper boxes; then,

$$T_{1,1} = 40; \; T_{1,2} = 100; \; T_{1,3} = 300;$$

We proceed to describe our problem as an integer program.

- Let S_p denote the subset of sub-suppliers that participate in the deal. As before, the total number of sub-suppliers of S_i is denoted by m_i.
- Denote by n_j the number of units required from item j, $1 \leq j \leq R$.
- Let $x_{i,l,j}$ be the number of units that $S_{i,l}$ provides from item j, $1 \leq j \leq R$.

Note that a few sub-suppliers of the same supplier may participate in the deal.

In the next example we give an input for which the optimal solution has this characteristic. First, we introduce the concept of *a package* that is often used in the multiple-item market.

Definition 3.3: A package is an *R-tuples* $(a_1, \ldots, a_R)$, in which a_j is the number of units supplied from item j. Let c_j denote the cost per unit of item j in the package; then, the price of the package is $\sum_{j=1}^{R} c_j a_j$.

Example 3.3: As in Example 3.2, suppose that the items are *printers* (item no. 1), *cartridges* (item no. 2) and *paper boxes* (item no. 3). There are two suppliers,

S_1 and S_2.

S_1 supplies the packages

(3,5,5) at the total cost of *395*;

(2,3,4) at the total cost of *350*.

(Thus, S_1 is represented by two sub-suppliers).

S_2 supplies the package *(5,10,10)* at the total cost of *750*.

Assume that $T_{1,1} = T_{2,1} = 7$, $T_{1,2} = T_{2,2} = 10$ *and* $T_{1,3} = T_{2,3} = 10$.

Suppose that we need to order *5* printers, *8* cartridges *and* *8* paper boxes.

An optimal deal would be to buy two packages from S_1: one package of *(3,5,5)* and one package of *(2,3,4)*.

As before, we denote by S_p the set of sub-suppliers that participate in the deal.

The following is the IP formulation of the RDS problem with multiple items:

$$\text{Minimize} \quad \sum_{i=1}^{m} \sum_{l=1}^{m_i} \sum_{j=1}^{R} c_{i,l,j} x_{i,l,j}$$

$$\text{Subject to} \quad \sum_{i=1}^{m} \sum_{l=1}^{m_i} x_{i,l,j} \geq n_j \quad \forall 1 \leq j \leq R$$

$$x_{i,l,j} \geq r_{i,l,j} \quad S_{i,l} \in S_p \quad \forall 1 \leq j \leq R$$

$$x_{i,l,j} \leq u_{i,l,j} \quad S_{i,l} \in S_p \quad \forall 1 \leq j \leq R$$

$$x_{i,l,j} = 0 \quad S_{i,l} \notin S_p \quad \forall 1 \leq j \leq R$$

$$\sum_{l=1}^{m_i} x_{i,l,j} \leq T_{i,j} \quad \forall 1 \leq j \leq R, \forall 1 \leq i \leq m$$

As in the single item case, we can solve this program optimally, by assigning first $r_{i,l,j}$ units of item j to the sub-supplier $S_{i,l} \in S_p$ and updating $u_{i,l,j} = u_{i,l,j} - r_{i,l,j}$, $T_{i,j} = T_{i,j} - r_{i,l,j}$. Then we can assign to each sub-supplier the maximal possible number of units from each item type, by using R lists, each sorted in non-decreasing order, by costs-per-unit. This procedure yields an optimal solution to the above IP.

Unless specified otherwise, in the following sections we do not assume either monotonicity or rationality on the price tables. In the next sections we refer to the general DS problem, in which the number of sub-suppliers participating in the deal may be arbitrarily large.

4. Greedy Approximation Algorithms

4.1 Single Item

4.1.1 The Residual Greedy Algorithm

We consider first a natural greedy algorithm, that attempts to satisfy in each stage a maximal fraction of the order at minimal cost. Formally, the Residual Greedy (RG) algorithm proceeds as follows. Let *need* denote the number of units that still need to be supplied, to satisfy the order. Initially, we set *need=n*. Now, *RG* keeps a list of sub-suppliers. In step s, $s \geq 1$,

1. Find the sub-supplier $S_{i,l}$ of some supplier $1 \leq i \leq m$, satisfying

$$r_{i,l} \leq need, \qquad (1)$$

such that $c_{i,l}$ is minimal.

2. Let $x_{i,l} = min\ (u_{i,l},\ need)$: buy from $S_{i,l}$ $x_{i,l}$ units of the item.

3. Update the maximal number of units that can be provided by S_i, that is,

$T_i = T_i - x_{i,l}$; update the set of sub-suppliers of S_i, omitting sub-suppliers $S_{i,l}$ for which $r_{i,l} > T_i$.

4. $need = need - x_{i,l}$; if $need > 0$ goto 1.

Example 4.1: Suppose that we have 4 suppliers, with the price tables given in Table 4.1. We need to supply at least 30 units of the item. Also, assume that

$T_1 = 50;\ T_2 = 40;\ T_3 = 5;$ and $T_4 = 6.$

	S_1	S_2	S_3	S_4
range	[1,5]	[1,4]	[1,2]	[1,2]
unit cost	34	32	31	35
range	[5,10]	[5,15]	[3,5]	[3,6]
unit cost	31	30	19	20
range	[11,50]	[16,20]		
unit cost	30	24		
range		[25,40]		
unit cost		21		

Table 4.1: A price table for Example 4.1 (the RG algorithm)

The algorithm RG operates as follows. Initially, it selects S_3, which supplies *5* units at the price of *19* per unit; then, it selects S_4, which supplies *6* units at the price of *20* per unit. Finally, it selects S_2, which supplies *17* units at the price of *24* per unit. The overall cost of the deal is *689*.

Note that we can get a better deal if we choose to buy *5* units from S_3, and the remaining units from S_2. We get that the overall cost is 515.

While RG is very simple to implement, it can perform poorly compared to the optimum, as stated in the next result.

Observation 4.1: *RG is an $\Omega(n)$ approximation for the DS problem with a single item, even when the price tables are monotone and rational.*

Proof: Consider the following instance. We need to buy at least n units of the item. The supplier S_0 can provide at least $(n+1)$ units at the cost of $c/(n+1)\geq 1$ per unit, while each of the suppliers, $S_1, \ldots, S_n$ can provide one unit of the item at the cost c. Now, OPT buys $n+1$ units from S_0 and pays the total cost c; RG buys a single unit from each of the suppliers $S_1, \ldots, S_n$, and overall pays $n \cdot c$. ■

4.1.2 A Better Greedy Algorithm

As we have shown, the algorithm RG, which considers in each stage only a subset of the sub-suppliers (those satisfying (1)), may be far from the optimal. Indeed, in some cases it may be better to buy more than *need* units, with the promise that the overall cost of the deal is minimized. The algorithm RG' that we describe below, allows to consider also sub-suppliers who can provide only *more* than *need* units.

1. Sort the sub-suppliers in non-decreasing order by the price per unit. Denote the sorted list by L, and let $r[j]$ ($u[j]$) and $c[j]$ denote the minimal (maximal) number of units provided by the sub-supplier $S[j]$, that is in position j in L, and the price per unit for this sub-supplier. Sub-suppliers with the same price per unit appear in the list in arbitrary order.

2. Let $need=n$;

3. $j=1$;

4. While $r[j] \leq need$ {buy from $S[j]$ $x[j]=min(need, u[j])$ units of the item; $need=need-x[j]$; *if* $need=0$ *then stop else* {Let S_i be the supplier such that $S[j]$ is a sub-supplier of S_i. Update the maximal number of units of the item available from S_i; that is, $T_i = T_i - x[j]$; update accordingly the entries of the set of sub-suppliers of S_i; if the sub-supplier $S[j]$ was omitted from L then $j = j+1$};

5. Let $C_{RG'}(need)$ be the overall cost of buying *need* units of the item, if we apply RG' recursively with $n = need$, starting from the first entry j^* in L, such that $j^* > j$, and $r[j^*] \leq need$; let C_f be the minimal possible total cost of buying (at least) *need* units from a single sub-supplier, $S[f]$, who can complete the deal.

   If $C_{RG'}(need) < C_f$ then {let $j^* = min\{j' > j: r[j] \leq need\}; j=j^*$; goto 4}

   else buy from $S[f]$ $r[f]$ units and stop.

Example 4.2: Suppose that we have 4 suppliers with price tables as given in Table 4.1. As before, assume that $T_1 = 50$, $T_2 = 40$, $T_3 = 5$ and $T_4 = 6$. Initially, we set $need=24$. First, RG' sorts the sub-suppliers in the table. The resulting list is:

$$L=\{([3,5],19),\ ([3,6],\ 20),([25,40],\ 21),([16,20],\ 24),\ ([5,15]),\ 30),\ ([11,50],\ 30)$$

$$([5,10],31),\ ([1,2],\ 31),\ ([1,4],\ 32),\ ([1,\ 5],\ 34),\ ([1,2],\ 35)\}$$

(For example, the first entry refers to the second sub-supplier of S_3, for which the minimal and maximal number of units are 3 and 5, respectively, and the price per unit is 19. This sub-supplier becomes $S[1]$).

RG' starts to scan the list L from left to right.

1) In the first step, RG' buys 5 units at the cost of 19 per unit from S[1].

Then we get that $need = need - 5 = 19$; $total = 19 \cdot 5 = 95$. Now, no units of the item are available from S_3; thus, $S[1]$ and $S[8]$ are omitted from L.

2) In the second step RG' buys 6 units from S[2], at the cost of 20 per unit.

We get that $need = need - 6 = 13$; $total = total + 20 \cdot 6 = 215$. Now, no units of the item are available from S_4; thus, $S[2]$ and $S[11]$ are omitted from L.

3) In the third step RG' finds that $r[3] > need$, then it computes

$$C_f = 16 \cdot 24 = 384 \text{ and } C_{RG'}(need) = 13 \cdot 30 = 390;$$

Since $C_{RG'}(need) > C_f$ RG' completes the deal, buying 16 units from $S[4]$, and $total = total + 384 = 599$.

Note that the minimal cost of the deal satisfies

$$Total_{OPT} \geq 19 \cdot 5 + 20 \cdot 6 + 13 \cdot 21 = 488.$$

4.2 Multiple Items

We now describe a Greedy algorithm for buying multiple items. Note that in or decomposition of the suppliers to sub-suppliers (as described in Section 3.2) we allow each sub-supplier to offer many packages, because each sub-supplier has ranges associated with the various items it supplies. Now we take a different approach: given the price tables defining the set of sub-suppliers, we generate the set of all the packages possible for each sub-supplier. This is done by explicit enumeration over each range in the table of the corresponding supplier.

To simplify our approximation algorithm, called Multi-Residual Greedy (MRG), we assume that each sub-supplier can be identified with a single package of the items.

It would be convenient to describe our minimization problem in terms of finding a minimum weight set cover. In the WEIGHTED SET COVER problem [GJ79] we have a set of items $U=\{ u_1, \ldots, u_n \}$, and a collection of subsets of the items $C= \{A_1, \ldots A_q\}$, where each subset, A_i, has a weight, $w(A_i)$. Our goal is to find a sub-collection of the subsets, $C' \subseteq C$, in which each of the items in U appears *at least* once and the total weight $\sum_{S \in C'} w(S)$ is minimized.

Given an order for items from R different types, in which we need n_j units from item j, we assume that U consists of R items.

Denote by *covered(j)* the fraction already covered from item j, and let *total* denote the overall cost of the deal. Initially, *total =0* and for all j, *covered(j) =0*.

For convenience we define also *need(j)=1-covered(j)*. We represent the set of possible packages by the collection of subsets $C= \{A_1, \ldots, A_q\}$: $w(A_i)$ is the cost of the package offered by A_i. Let $f_{A_i}(j)$ denote the fraction of item j covered by A_i. The algorithm computes the value α_{A_i} for each A_i (see in Step 3.): α_{A_i} measures the average cost per unit of an item, when provided by A_i. The average is taken over the individual contributions of A_i to covering the demands of the items 1,..., R. (For example, if R=2 and A_i covers ½ of the demand for item 1 and ¾ of the demand for item 2, then overall A_i covers *5/4* items out of 2). Note that we treat items of different types uniformly, and we only count the total amount of items that still need to be covered.

The algorithm MRG proceeds as follows.

```
1.  For $j=1,\ldots,R$ $covered(j)=0;$ $need(j)=1.$
2.  $total=0;$
3.  While for some $1 \le j \le R$, $covered(j) < 1$ do
      For each subset, $A_i \in C$, let $f'_{A_i}(j) = min(f'_{A_i}(j), need(j))$; compute

          $\alpha_{A_i} = \frac{w(A_i)}{\sum_{j=1}^{R} f_{A_i}(j)}$ ;

      Select the set $A_i$ for which $\alpha_{A_i}$ is minimal. Denote this set by $A_{min}$.
      For any item $1 \le j \le R$,
          if $A_{min}$ contributes to the covering of $j$ the fraction $f_{A_{min}}(j) > 0$
            then $covered(j) - covered(j) + f_{A_{min}}(j)$
      $total = total + w(A_{min})$
      Update the number of available items of type $1 \le j \le R$ for each supplier.
      Omit from $C$ subsets corresponding to packages of suppliers who have
      already used all the available units from some item.
4.  Output the picked subsets.
```

The next example shows the operation of the algorithm MRG.

Example 4.3: Suppose that there are two suppliers, S_1, S_2 and three items: printers, cartridges and paper. In the following we give the price tables for S_1, S_2 which describe the packages that can be provided by the two suppliers.

	printers	cartridges	paper

range	[1,1]	[3,3]	[4,4]
unit cost	300	30	15
range	[5,5]	[10,10]	[15,15]
unit cost	250	23	10

Table 4.2: The price table of S_1.

Thus, for example, the price of the package including one printer, *3* cartridges and *4* paper boxes is *450*.

	printers	cartridges	paper
range	[1,1]	[5,5]	[8,8]
unit cost	305	27	16
range	[8,8]	[20,20]	[30,30]
unit cost	260	24	9

Table 4.3: The price table of S_2.

Suppose that we need *12* printers, *20* cartridges and *36* boxes of paper. S_1, S_2 can provide up to *30* printers, *100* cartridges and *150* boxes of paper. Hence,

$$T_{1,1} = T_{2,1} = 30,\ T_{1,2} = T_{2,2} = 100 \text{ and } T_{1,3} = T_{2,3} = 150.$$

Let

$$A_1=(1,3,4);\ A_2=(5,10,15);\ A_3=(1,5,8);\ A_4=(8,20,30)$$

With the corresponding weights

$$w(A_1)= 450;\ w(A_2)=1630;\ w(A_3)=568;\ w(A_4)= 2830;$$

Let $C_1=\{A_1,A_2\}$ and $C_2=\{A_3,A_4\}$. Our collection of sets is $C=\{C_1,C_2\}$.

We describe the operation of the algorithm MRG by iterations.

(i) In the first iteration we have *need(1)=need(2)=need(3)=1*;

Now we compute $\alpha_{A_1},\ldots,\alpha_{A_4}$.

$$\alpha_{A_1}=\frac{450}{1/12+3/20+4/36}=1306.45$$

Similarly, we get that $\alpha_{A_2}=1222.5$; $\alpha_{A_3}=1022.4$ and $\alpha_{A_4}=1132$;

Therefore MRG selects $A_{min}=A_3$, and updates

covered(1)=1/12; covered(2)=5/20; covered(3)–8/36;

and *total=568;*

In the second iteration we have

$\alpha_{A_1}=1306.45$; $\alpha_{A_2}=1222.5$ and $\alpha_{A_3}=1022.4$;

Now, when computing α_{A_4}, the set A_4 contributes to item 2 only *15/20* and to item 3 only *28/36*, as *8/36* are already covered; therefore,

$$\alpha_{A_4}=\frac{2830}{8/12+15/20+28/36}=1289.62$$

MRG selects again A_3 and we get that

covered(1)=2/12; covered(2)=10/20; covered(3)=16/36;

and *total=1136;*

In the third iteration A_3 is selected again and we have

covered(1)=3/12; covered(2)=15/20; covered(3)=24/36;

and *total=1704;*

A_3 is selected again. Now we have

covered(1)=4/12; covered(2)=20/20; covered(3)=32/36;

Hence,

need(1)=8/12; need (2)=0; need (3)=4/36;

and *total=2272;*

(ii) We compute again,

$$\alpha_{A_1} = \frac{450}{1/12+0+4/36} = 2314.29$$

and similarly

$$\alpha_{A_2} = 3088.42;\ \alpha_{A_3} = 2921.15 \text{ and } \alpha_{A_4} = 3638.57;$$

Therefore MRG selects $A_{min} = A_1$, and updates

$$covered(1)=5/12;\ covered(2)=1;\ covered(3)=1;$$

and $total=2722;$

In the sixth iteration we find that

$$\alpha_{A_1} = 5400;\ \alpha_{A_2} = 3912;\ \alpha_{A_3} = 6816 \text{ and } \alpha_{A_4} = 4851.43;$$

Therefore MRG selects $A_{min} = A_2$, and updates

$$covered(1)=10/12;\ covered(2)= covered(3)=1;$$

and $total=4352;$

In the seventh iteration we have

$$\alpha_{A_1} = 5400;\ \alpha_{A_2} = 9780;\ \alpha_{A_3} = 6816 \text{ and } \alpha_{A_4} - 16980;$$

Therefore MRG selects $A_{min} = A_1$.

In the last iteration MRG selects again $A_{min} = A_1$. We get that $total=5252$.

Note that MRG is not optimal: a better deal is to select once A_1 and then A_4, at the total cost 4460.

4.2.1 MRG versus RG

While it may seem initially that RG is simply MRG in the special case where

$R=1$, we now show, that the two algorithms may behave differently.

Example 4.4: Suppose that we have a single item and three suppliers, S_1, S_2 and S_3. We need to provide at least *30* units of the item at minimal total cost.

Each of the suppliers S_1, S_2 can provide 100 units and S_3 10 units.

S_1	S_2	S_3
[1,10]	[1,20]	[1,10]
30	35	12
[11,40]	[21,50]	
25	15	

Table 4.4: The price table for Example 4.4.

Now, RG initially chooses to buy *10* units from S_3, at the price of *12* per unit.

Then we get that *need=20*, therefore RG next selects S_1, which supplies *20* units at the cost of *25* per unit. The total cost is *620*.

In contrast, MRG maintains a list with all the possible deals. First, MRG selects to buy from S_3 *10* units, at the cost of *12* per unit. Then, MRG buys *21* units from S_2, and the total cost is *435*.

4.2.2 Implementation of MRG

Note that algorithm MRG uses as input the set of all possible packages offered by each of the sub-suppliers. Indeed, when the input is given as price tables, for generating the A_i's we need to enumerate all the possible packages for each sub-supplier. In generating the A_i's and computing the α_{A_i} *'s* we may wish to consider the following simplifications:

- Since our goal in each iteration is to find A_{min}, we can save space by generating the A_i's dynamically and maintaining the minimum in each step. This, however, requires to recompute in each iteration for A_i the values $f_{A_i}(j)$, for $j=1,\ldots,R$.

- At the beginning of each iteration it may help to compute first the value α_{A_i} for the set, A_i that was selected in the previous iteration to be A_{min}. If α_{A_i} remains unchanged then the same set remains A_{min} in this iteration. This is due to the fact that for any other set α_{A_i} cannot decrease when moving to the next iteration.

- When the A_i's are derived from the price tables of the suppliers, we can avoid the enumeration of all the packages of a given sub-supplier, $S_{i,l}$, as follows. Let $needu(j)=need(j)\cdot n_j$. We set $\hat{u}_{i,l,j}=max(min(u_{i,l,j},needu(j)),r_{i,l,j})$. Let

$$g_j(n_1,\ldots,n_R)=\begin{cases} 1 & n_j \geq needu(j) \\ \dfrac{n_j}{needu(j)} & otherwise \end{cases}$$

and let

$$f_{i,l}(n_1,\ldots,n_R)=\frac{\sum_{j=1}^{R} c_{i,l,j}\cdot n_j}{g(n_1,\ldots,n_R)}$$

Now we find the integral point $(n_1,\ldots,n_R)$, $r_{i,l,j}\leq n_j\leq \hat{u}_{i,l,j}$, in which $f_{i,l}(\cdot)$ gets the minimum. We take the resulting package, A_l, to be the package representing the sub-supplier $S_{i,l}$. Thus, in each iteration of the algorithm we need to consider at most $\sum_i m_i$ packages.

5. Approximation Schemes for Single Item

In this section we present algorithms that achieve a $(1+\varepsilon)$-approximation for the DS problem for single-item deals, for a given $\varepsilon>0$. In §5.1 we discuss an easy variant of our problem, where each of the suppliers can provide an unbounded number of units from the item. In §5.2 we refer to the bounded case. For both cases we develop *approximation schemes*, which yield pseudo-polynomial time optimal algorithms. We analyze these algorithms and show how their running time can be reduced when the price tables are rational and monotone.

Our algorithms use as procedures fully polynomial time approximation schemes *(FPTAS)* for variants of the 0-1 knapsack problem. Since we use the cost of partial deals as the "sizes" of the items in the resulting packing problems, it is implicitly assumed that the entries in the price tables are integers. This does not impose a restriction on the inputs for the DS problem. We allow the costs per unit to be any positive rationals. Before we apply our approximation schemes, we can use the following ***rounding procedure:***

1. Given a set of price tables $T_1, \ldots, T_m$, for the suppliers $S_1, \ldots, S_m$, write each entry as a rational number.
2. Find the least common denominator, D, of all entries.
3. Multiply all entries by D.

Thus, we get that all entries become integral.

.1 Unbounded Case

Assume first that we can repeatedly buy from some sub-supplier, $S_{i,l}$, i.e., for any $l \leq i \leq m$, $T_l \geq n$, where n is the number of units ordered from the item. The algorithm is based on an FPTAS for the INTEGER KNAPSACK problem [GJ-79]

(also known as the Unbounded Knapsack [L-79]):

Input: A set of items, U, where each item $u \in U$ has the size $s(u) \in Z^+$ and the value $v(u) \in Z^+$, and positive integers, B, K.

Question: Is there an assignment of a non-negative integer $c(u)$ to each item $u \in U$, such that $\sum_{u \in U} s(u)c(u) \leq B$, and $\sum_{u \in U} v(u)c(u) \geq K$?

The problem is known to be NP-hard and solvable in pseudo-polynomial time (see, e.g., in [GL-79], [L-79]). In particular, Lawler developed in [L-79] an

FPTAS, whose running time is $O(|U|+1/\varepsilon^3)$, for any $\varepsilon>0$, where $|U|$ is the number of items in the input. This yields an optimal pseudo-polynomial time algorithm.

Denote by $A_\varepsilon(I)$, $OPT(I)$ the values of the approximate and optimal solutions for a given input, I, of INTEGER KNAPSACK. Let $V=max_{u\in U}v(u)$ be the maximal profit of any item in U. Then, $OPT(I)\leq B\cdot V$, since we can take at most B copies of the item whose profit is maximal. Hence, if we choose $\varepsilon=(B\cdot V)^{-1}$, and run some approximation scheme A_ε on an instance I, we get that

$$A_\varepsilon(I)\geq(1-\varepsilon)\ OPT(I),$$

or

$$1\geq\varepsilon\cdot OPT(I)\ \geq OPT(I)-A_\varepsilon(I)$$

and since the solution for INTEGER KNAPSACK is integral, we get an optimal solution. The running time of A_ε is $O(|U|+B^3V^3)$.

We describe below an approximation scheme for the unbounded DS problem. The idea is to define *building blocks* of sizes $1\leq s\leq n_0$ for some $n_0\geq 1$ (defined below), and to find the combination of blocks that yields the minimal total cost (Since we refer to the unbounded case, we allow to take *any* number of blocks of each type). The input parameter, ε, can get any value in the range $(0,1)$. The scheme proceeds as follows.

1. Run algorithm RG on the input instance, to obtain an upper bound for the minimal cost, C_{min}. Initially, set $C_{min}=C_{RG}$.

2. Let $\hat{c}$ be the minimal unit cost offered by any sub-supplier. For a given value of C_{min}, let $n_0=C_{min}/\hat{c}$ be the maximal possible number of units bought by an optimal algorithm, OPT.

3. Let *cost(s)* be the minimal cost of buying s units of the item from a single sub-supplier, $1\leq s\leq n_0$. (It may be the case that it is impossible to buy s units of the item from a single sub-supplier, for some values of s. For these values we set $cost(s)=C_{min}+1$).

4. For each pair of values (C_{min}, n_0), generate an instance of the INTEGER KNAPSACK problem: the universe, U, consists of n_0 items, $u_1, \ldots, u_{n_0} \in U$, such that $s(u_s) = cost(s)$ and $v(u_s)=s$, $1 \leq s \leq n_0$. Given $\varepsilon>0$, find the maximal profit from packing items in U in a knapsack of capacity $B = C_{min}$. (This can be done by using, e.g., the FPTAS for INTEGER KNAPSACK given in [L-79]).

5. Use a binary search to find the minimal value of C_{min}, $1 \leq C_{min} \leq C_{RG}$, which yields a feasible solution, i.e., the profit (= number of items provided) is at least n.

For the time complexity of the above scheme, we first note that for each guess of C_{min}, for which we define $n_0 = C_{min}/\hat{c}$, and for any $\varepsilon>0$, we can get a $(1+\varepsilon)$-approximation for the resulting instance of INTEGER KNAPSACK in $O(n_0+1/\varepsilon^3)$ steps [L-79]. Hence, we get that the complexity for each guess of

C_{min} is

$$O(C_{min}+ 1/\varepsilon^3) = O(C_{RG} + 1/\varepsilon^3).$$

Now, the number of guesses of C_{min} *equals to* $\log_{1+\varepsilon} C_{RG}$, *hence, for a given value of* C_{RG}, *the running time of the scheme is*

$$O(\log_{1+\varepsilon} C_{RG} (C_{RG} + 1/\varepsilon^3)).$$

Denote by c_{max} *the maximal unit price for any sub-supplier, then* $C_{RG} \leq n \cdot c_{max}$, *and the overall time complexity is*

$$O(\log_{1+\varepsilon} (n \cdot c_{max})(n \cdot c_{max} + 1/\varepsilon^3)).$$

Finally, recall that the number of sub-suppliers of supplier i is m_i; the running time of RG is linear in the total number of sub-suppliers, given by $M=\sum_i m_i$

(2)

Thus, overall, we get that the running time of the scheme is

$$O(\log_{1+\varepsilon} (n \cdot c_{max})(n \cdot c_{max} + 1/\varepsilon^3)+ M).$$

When the price tables are rational and monotone, we can reduce the number of elements in the instance of INTEGER KNAPSACK to $\log_{1+\varepsilon} n_0$. Suppose that for some $1 \le s \le n_0$, the supplier that can provide s units of the item at minimal cost is S_h, for some $1 \le h \le m$. For $1 \le s < s' \le (1+\varepsilon)s$, let c_u, c'_u be the cost per unit, for buying s and s' units from S_h, respectively. Since the table of S_h is rational,

$$c'_u \cdot s' = cost_h(s') \ge cost_h(s) = c_u \cdot s$$

Hence,

$$\frac{c_u}{c'_u} \le \frac{s'}{s} \le 1+\varepsilon$$

In addition, since the table is monotone, $c_u \ge c'_u$. It follows, that

$$cost_h(s') \le c_u \cdot s' = c_u \cdot (s'/s) \cdot s \le (1+\varepsilon)\, cost_h(s)$$

Hence, instead of taking in the instance of INTEGER KNAPSACK all the values $1 \le s \le n_0$, we can take only values of s which are (rounded) integral powers of $(1+\varepsilon)$. From the above discussion, it is guaranteed, that if

$(1+\varepsilon)^{r-1} < s \le (1+\varepsilon)^r$, and we buy from the supplier S_h $\lfloor (1+\varepsilon)^r \rfloor$ units, then the total cost of the deal may increase by at most factor $(1+\varepsilon)$.

Thus, when the price tables are monotone and rational, the complexity of solving the INTEGER KNAPSACK is

$$O((\log_{1+\varepsilon} n_0)+1/\varepsilon^3) = O(\log_{1+\varepsilon} C_{RG} + 1/\varepsilon^3)$$

As before, the number of guesses is $\log_{1+\varepsilon} C_{RG}$. Hence, the overall complexity is

$$O((\log_{1+\varepsilon} C_{RG})^2 + \log_{1+\varepsilon} C_{RG}/\varepsilon^3 + M)$$

$$= O((\log_{1+\varepsilon}(n \cdot c_{max}))^2 + \log_{1+\varepsilon}(n \cdot c_{max})/\varepsilon^3 + M)$$

5.2 Bounded Case

Now, we describe an approximation scheme for the case where each supplier, S_i, has a *limited* number of units from the item, given by T_i, $1 \leq i \leq m$. As before, we use *building blocks*, that will represent the deals in which we buy all the items from a single supplier; thus, the number of "blocks" for a supplier S_i is at most T_i, $1 \leq i \leq m$. Also, to guarantee that S_i provides at most T_i units from the item, we assign to each supplier, S_i, at most *one* "building block", whose size is in the range $[1, T_i]$.

We assume in the analysis of our scheme, that we can find in $O(1)$ steps the minimal cost of buying s units of the item from S_i. Clearly, this holds when the price tables are rational and monotone; for general tables, some preprocessing may be required. As in the unbounded case, the input parameter, ε, can get any value in the range $(0,1)$.

The algorithm is based on an FPTAS for the 0-1 MULTIPLE CHOICE KNAPSACK problem (MCK) [GJ-79]:

Input: A universe U of n items, partitioned into m sets, $U_1, \ldots, U_m$, and the positive integers, B, K. There are k_i items in U_i, $\sum_{i=1}^{m} k_i = n$; s_{ij} and v_{ij} are the size and the value, respectively, of the j-th item in U_i, $1 \leq j \leq k_i$.

Question: Is there an assignment of $x_{ij} \in \{0,1\}$, for all i,j, such that $\sum_{i=1}^{m} \sum_{j=1}^{k_i} s_{ij} x_{ij} \leq B$, and $\sum_{i=1}^{m} \sum_{j=1}^{k_i} v_{ij} x_{ij} \geq K$, and for all $1 \leq i \leq m$ $\sum_{j=1}^{k_i} x_{ij} \leq 1$?

The problem is known to be NP-hard and solvable in pseudo-polynomial time (see, e.g., in [GJ-79], [L-79]). We use in our analysis the FPTAS presented by

Lawler [L-79], whose running time is $O(n \log n + mn/\varepsilon)$, for any $\varepsilon > 0$.

Our scheme proceeds as follows.

1. Run algorithm RG on the input instance, to obtain an upper bound for the minimal cost, C_{min}. Initially, set $C_{min} = C_{RG}$.

2. Let $T = \sum_{i=1}^{m} T_i$ denote the total number of units of the item available from all the suppliers. For each value of C_{min}, generate the following instance of the MCK problem. The universe, U, consists of $n=T$ items, partitioned to m sets $U_1, \ldots, U_m$. The set U_i represents the supplier S_i; the size of U_i is T_i, the maximal number of units

available from S_i. For the j-th item in U_i, s_{ij} is the minimal cost of buying j units from S_i, and $v_{ij} = j$. Now, we find the maximal profit from packing items from $U_1, \ldots, U_m$ in a knapsack of capacity $B=C_{min}$, with the restriction, that from each set we choose at most one item.

3. Use a binary search to find the minimal value of C_{min}, $1 \le C_{min} \le C_{RG}$,

which yields a feasible solution, i.e., the profit (= number of items provided) is at least n.

For the time complexity of the above scheme, we first note that for each guess of C_{min}, and for any $\varepsilon>0$, we can get a $(1+\varepsilon)$-approximation for the resulting instance of MCK in $O(T \log T + mT/\varepsilon)$ steps [L79]. Now, the number of guesses of C_{min} equals to $\log_{1+\varepsilon} C_{RG}$, hence, for a given value of C_{RG}, the running time of the scheme is

$$O(\log_{1+\varepsilon} C_{RG} (T \log T + mT/\varepsilon))$$

$$= O(\log_{1+\varepsilon} (n \cdot c_{max}) (T \log T + mT/\varepsilon))$$

Note, that since $T_i \ge m_i$, for $1 \le i \le m$, adding the running time of RG ($=\sum_{i=1}^{m} m_i$) does not affect the overall complexity of the scheme.

When the price tables are rational and monotone, we can reduce the number of elements in the instance of MCK to $T' = \sum_{i=1}^{m} \lceil \log_{1+\varepsilon} T_i \rceil$: from each supplier, S_i, we allow to buy number of units, which is an integral power of $(1+\varepsilon)$. This may increase the overall cost of the deal at most by factor $(1+\varepsilon)$, as argued in Section 5.1.Thus, the number of items in the set U_i is at most $\lceil \log_{1+\varepsilon} T_i \rceil$; s_{ij} is the cost of buying $(1+\varepsilon)^j$ units from S_i, and $v_{ij} = (1+\varepsilon)^j$. Now, we add also the running time of RG, and get that the overall running time of the scheme is

$$O(\log_{1+\varepsilon} (n \cdot c_{max}) (T' \log T' + mT'/\varepsilon) + M),$$

Let $T_{max} = max_i\, T_i$, then $T' = O(m \log T_{max})$, and we get that the overall complexity is

$$O(\log_{1+\varepsilon}(n\cdot c_{max})(m(\log T_{max})^2 + m^2 \log T_{max} / \varepsilon)+M).$$

6. Approximation Schemes for Multiple Item Deals

Assume now that the unit cost for each item is given in various ranges, and that each supplier offers *combinations* of the appropriate number of units, from the R items, in each of the ranges (see e.g. in *Table 3.2*).

6.1 Unbounded Case

We assume first that the number of units of each item is unbounded, for any supplier. For this case we propose an approximation scheme that uses as input the set of all possible *packages* offered by the suppliers. We use as procedure an approximation scheme for a variant of the Integer Multi-Dimensional Knapsack problem [CHW-76]. (We describe this problem in detail below). Our general scheme proceeds as follows.

1. Run algorithm MRG on the input instance, to obtain an upper bound for the minimal cost, C_{min}. Initially, set $C_{min} = C_{MRG}$.

2. For a given value of C_{min}, scale the package prices as follows: round up the price of each package to the nearest multiple of $\varepsilon \cdot C_{min}/m$, where m is the number of suppliers.

   Divide by $\varepsilon \cdot C_{min}/m$ the prices of all packages, such that all prices are in the range $\{0, \ldots, m/\varepsilon\}$. Finally, round up the price of each package to the nearest integral power of $(1+\varepsilon)$. Now we have $O(\ln m)$ price categories for the packages.

3. For group l (i.e., the price of a package is $(1+\varepsilon)^l$) find a vector of length R, given as a set of integral values, $1 \leq \alpha_j \leq 1/\varepsilon$, describing the amount of items of each type covered by that group. More specifically, if group l covers the vector $(a_1, \ldots, a_R)$ then the packages selected for the deal will cover at least $b_j = a_j \cdot \varepsilon n_j$ units from item j, $1 \leq j \leq R$.

4. Given a covering vector for group l, find a subset of packages in this

group that covers the vector at minimal cost.

5. Use a binary search to find the minimal value of C_{min}, $1 \leq C_{min} \leq C_{MRG}$, which yields a feasible solution, i.e., the number of units provided from item j, for all $1 \leq j \leq R$ is at least n_j.

We now describe in detail Step 4. of the scheme. In this step we need to find a combination of packages of a given cost category, which covers certain amount of items from each of the R types, at minimum cost. We use the following LP relaxation of our problem. We call this problem SUP. Suppose that there are N_l packages in group l, $1 \leq N_l \leq N$. Given the non-negative rational values c_i, b_j and a_{ij}, where $1 \leq i \leq N_l$, and $1 \leq j \leq R$, we need to solve the following linear program.

$$\text{Minimize} \quad \sum_{i=1}^{N_l} c_i x_i$$

$$\text{Subject to} \quad \sum_{i=1}^{N_l} a_{ij} x_i \geq b_j \quad j=1,\ldots,R$$

$$x_i \geq 0 \quad i=1,\ldots,N_l$$

For such a system of inequalities there is a solution in which at most R values, $x_{i_1},\ldots,x_{i_R}$ get non-zero values (see e.g. in [L-76]). Hence, it suffices to solve the above linear program for the $\binom{N_l}{R}$ possible subsets of R variable out of $(x_1,\ldots,x_{N_l})$.

We now describe an approximation scheme, based on an optimal solution for the above program.

Algorithm Multi-dimensional Cover (MDC): Let $x_{i_1},\ldots,x_{i_R}$ be an optimal solution for the problem SUP. We take $\lceil x_{i1} \rceil,\ldots,\lceil x_{iR} \rceil$ as approximate solution.

Denote by C_{SUP} the optimal cost for the problem SUP; C_{MDC} is the cost of algorithm MDC and C_0 is the optimal cost for our original covering problem (in which we can take an integral number of units from each package). We denote by

$c_{i_1}, \ldots, c_{i_u}$ the costs of the packages that are supplied in the optimal solution for SUP. Note that $C_o \geq C_{SUP}$. Hence we get that

$$C_{MDC} - C_o \leq C_{MDC} - C_{SUP} \leq c_{i_1} + \cdots + c_{i_k} \leq R \cdot max\{c_1, \ldots, c_{N_l}\}.$$

We now describe an $(1+\varepsilon)$-approximation scheme.

Given a vector x we sort the entries such that $c_1 \geq \cdots \geq c_{N_l}$. For a given $\varepsilon > 0$, let $\varepsilon' = \varepsilon/(1+\varepsilon)$, and $\delta = \lceil k \cdot ((1/\varepsilon') - 1) \rceil$. Denote by Ω the set of integer vectors $x = (x_1, \ldots, x_{N_l})$ satisfying $x_l \geq 0$ and $\sum_{l=1}^{N_l} x_i \leq \delta$.

For any vector $x \in \Omega$ we run algorithm MDC for the following problem.

Let $d \geq 1$ be the maximal integer i for which $x_i \neq 0$. Then we solve the LP for the problem SUP, given by

$$\text{Minimize} \quad \sum_{i=d+1}^{N_l} c_i z_i$$

$$\text{Subject to} \quad \sum_{l=d+1}^{N_l} a_{ij} z_i \geq b_j - \sum_{l=1}^{N_l} a_{ij} x_i \quad j=1,\ldots,R$$

$$z_i \geq 0 \quad i=1,\ldots,N_l$$

Denote by $C_{MDC}(x)$ the value obtained from running MDC on the fractional solution of the above program, with a given vector x, and let $c(x) = \sum_{i=1}^{N_l} c_i x_i$. The algorithm MDC$_\varepsilon$ selects the vector x for which

$$C_{MDC_\varepsilon}(x) = min_{x \in \Omega} \left(c(x) + C_{MDC}(x) \right)$$

Using arguments similar to those in [CHW-76] it can be shown that (i) The running time of algorithm MDC$_\varepsilon$ is $O(N^{\lceil R/\varepsilon' \rceil})$, and its space complexity is $O(N)$; (ii) If $C_o \neq 0, \infty$ then $C_{MDC_\varepsilon}/C_o < 1 + \varepsilon$.

We now compute the overall time complexity of the above scheme. The running time of the scheme consists of the running time of MRG, to which we add Steps 3.– 5. The heart of the scheme is Step 4. For a given value of C_{min} we run algorithm MDC$_\varepsilon$, for each cost group l, $1 \leq l \leq log_{1+\varepsilon}(m/\varepsilon)$ taking all the possible allocation vectors $(\alpha_1, \ldots, \alpha_R)$. We define a *configuration* to be a given set of allocation vectors for *all* the cost groups. The running time of Step 4. for each configuration is $O(log_{1+\varepsilon}\, n \cdot N^{\lceil R/\varepsilon \rceil})$, and the possible number of configurations is

$$O((m/\varepsilon)^{R \cdot ln(1/\varepsilon)/\varepsilon}).$$

Hence, for a given value of C_{min} the running time of Step 3.

and 4. is

$$O(log_{1+\varepsilon}\, n \cdot N^{\lceil R/\varepsilon \rceil} \cdot (\frac{m}{\varepsilon})^{R \cdot ln(1/\varepsilon)/\varepsilon}).$$

As before, we need to multiply the above complexity by the number of guesses of the value of C_{min}, which equals to $log_{1+\varepsilon}\, (n \cdot c_{max})$ and add the running time of algorithm MRG, which is linear in the total number of packages. This gives the overall running time of

$$O(N + log_{1+\varepsilon}\, (n \cdot c_{max}) \cdot log_{1+\varepsilon}\, n \cdot N^{\lceil R/\varepsilon \rceil} \cdot (m/\varepsilon)^{R \cdot ln(1/\varepsilon)/\varepsilon}).$$

6.2 The Bounded Case

In the bounded case we associate each supplier S_h with a set of packages, $\{i_{h,1},\dots,i_{h,N_h}\}$, where N_h is the number of distinct packages offered by S_h,

$1 \leq h \leq m$. Denote now by T_h^j the total number of units of item j held in the stock of S_h. We describe below an approximation scheme, which uses Steps 1. – 5. as in §7.1. However, we need to slightly modify algorithm MDC$_\varepsilon$ (in Step 4.).

We use in our scheme the following assumptions: (i) The number of suppliers is a fixed (but arbitrary) constant. (ii) There exists an optimal (possibly fractional) deal, where the number of units bought from each item, for any supplier h is at most $T_h^j/2$. Assumption (ii) implies that the order sizes are small relative to the amount of units available of the items, from each supplier. Thus, we refer here to *small customers*.

Our scheme for the bounded case proceeds similar to the scheme described in §7.1. In Step 4., we need to find a deal of minimum cost that covers the allocation vector assigned to group l. While doing so, we also need to verify that the overall number of units covered in our solution is bounded by T_h^j for all $1 \leq j \leq R$ and $1 \leq h \leq m$. This is done by allocating to each cost group l some fraction of the stock of item j supplied by S_h. We define for group l a stock vector of length $R \cdot m$, $\beta=(\beta_{1,1}, \dots, \beta_{1,R}, \dots, \beta_{m,1}, \dots, \beta_{m,R})$, where $1 \leq \beta_{h,j} \leq 1/(2\varepsilon)$. We select the set of stock vectors for the cost groups, such that the total number of units that can be allocated from item j by S_h is at most $T_h^j/2$. By Assumption (ii) there exists an optimal solution that uses at most half of the stock of item j, for each supplier. The stock allocated to the group l by S_h from item j is then given by $t_{h,j}=\beta_{h,j} \cdot \varepsilon\ T_h^j$.

Note that in this partition of the stock of each item for specific supplier, we allow non-integral number of units to be supplied by some groups. These non-integral values are used in the LP, and are later rounded by algorithm MDC.

We summarize below the modified scheme.

1. Run algorithm MRG on the input instance, to obtain an upper bound for the minimal cost, C_{min}. Initially, set $C_{min} = C_{MRG}$.

2. For a given value of C_{min}, scale the package prices as follows. Round up the price of each package to the nearest multiple of $\varepsilon \cdot C_{min}/N$, where N is the overall number of packages offered by the suppliers; divide by $\varepsilon \cdot C_{min}/N$ the prices of all packages, such that all prices are in the range $\{0, \dots, N/\varepsilon\}$; round up the price of each package to the nearest integral power of $(1+\varepsilon)$.

3. For group l (i) find a vector α of length R, given as a set of integral values, $1 \leq \alpha_j \leq 1/\varepsilon$, describing the amount of items of each type covered by that group; if group l covers the vector $(\alpha_1, \dots, \alpha_R)$ then the packages selected for the deal will cover at least $b_j - \alpha_j \cdot \varepsilon n_j$ units from item j, $1 \leq j \leq R$. (ii) Find a vector β of length Rm, given as a set of integral values, $1 \leq \beta_{h,j} \leq 1/(2\varepsilon)$, describing the amount of items of type j allocated to group l from the stock of the supplier S_h.

4. Given a covering vector α and a stock vector β for group l, find a subset of packages in this group that covers α while satisfying the stock restrictions given by β, at minimal cost.

5. Use binary search to find the minimal value of C_{min}, $1 \leq C_{min} \leq C_{MRG}$, which yields a feasible solution.

We now describe in detail Step 4. of the modified scheme. Define the set of vectors Ω and δ as in §7.1. For any vector $x \in \Omega$ we run algorithm MDC for the following problem.

Let $d \geq 1$ be the maximal integer i for which $x_i \neq 0$. Then we solve for group l the LP:

$$\text{Minimize} \quad \sum_{i=d+1}^{N_l} c_i z_i$$

$$\text{Subject to} \quad \sum_{i=d+1}^{N_l} a_{ij} z_i \geq b_j - \sum_{i=1}^{N_l} a_{ij} x_i \quad j=1,\ldots,R$$

$$\sum_{i=d+1}^{N_l} a_{ij} z_i \leq t_{h,j} - \sum_{i=1}^{N_l} a_{ij} x_i \quad j=1,\ldots,R \quad h=1,\ldots,m$$

$$z_i \geq 0 \quad i=1,\ldots,N_l$$

Denote by $C_{MDC}(x)$ the value obtained from running MDC on the fractional solution of the above program, with a given vector x, and let $c(x)=\sum_{i=1}^{N_l} c_i x_i$. The algorithm B-MDC$_\varepsilon$ selects the vector x for which

$$C_{B-MDC_\varepsilon}(x) = \min_{x \in \Omega} \left(c(x) + C_{MDC}(x) \right)$$

As before, we can use arguments similar to those in [CHW-76] to show that (i) The running time of algorithm MDC$_\varepsilon$ is $O(N^{\lceil Rm/\varepsilon \rceil})$, and its space complexity is $O(N)$; (ii) If $C_0 \neq 0, \infty$ then $C_{B-MDC_\varepsilon} / C_0 < 1 + \varepsilon$.

We now compute the overall time complexity of the modified scheme.

As in §7.1, the running time of the scheme consists of the running time of MRG,

to which we add Steps 3.– 5. For a given value of C_{min} we run algorithm

B-MDC$_{\varepsilon}$ for each cost group l, $1 \leq l \leq \log_{1+\varepsilon}(N/\varepsilon)$, taking all the possible allocation vectors $(\alpha_1, \ldots, \alpha_R)$ and stock vectors $(\beta_{1,1}, \ldots, \beta_{m,R})$. We define a *configuration* to be a given set of allocation and stock vectors for all the cost groups. The running time of Step 4. for each configuration is $O(\log_{1+\varepsilon} n \cdot N^{\lceil Rm/\varepsilon \rceil})$, and the possible number of configurations is $O((N/\varepsilon)^{Rm \cdot \ln(1/\varepsilon)/\varepsilon})$. Hence, for a given value of C_{min} the running time of Steps 3. and 4. is

$$O(\log_{1+\varepsilon} n \cdot N^{\lceil Rm/\varepsilon \rceil} \cdot (\frac{N}{\varepsilon})^{R \cdot m \cdot \ln(1/\varepsilon)/\varepsilon}).$$

Finally, we multiply the above complexity by the number of guesses of the value of C_{min}, which equals to $\log_{1+\varepsilon}(n \cdot c_{max})$, and add the running time of algorithm MRG. This gives the overall running time of

$$O(N + \log_{1+\varepsilon}(n \cdot c_{max}) \cdot \log_{1+\varepsilon} n \cdot (\frac{N}{\varepsilon})^{R \cdot m \cdot \ln(1/\varepsilon)/\varepsilon}).$$

Deal Splitting for Quantity-additive Deals

Terminology and Definitions

We start by introducing new concepts.

DS Algorithm

A DS algorithm accepts a collection of sellers and a request from a buyer, and finds a purchasing deal that satisfies the buyer's request with a minimum price.

Supplier, Seller

Each *seller* is a collection of *sub-sellers*, such that each sub-seller defines a collection of *items* that must be bought, together, within the ranges of their respective quantities. The *seller* also has an *availability vector* of quantity in stock, per each item.

Notes:

1. So far we used the terms Suppliers, and their respective Sub-Suppliers, whereas the Utility-Layer sections mention the term Seller; this section considers Sellers and Suppliers as synonyms, and similarly for the terms Sub-Supplier and Sub-Seller (see below).
2. A human trader, in light of the received buyer's RFQ, may define a sub-seller explicitly. Furthermore, a collection of sub-sellers may be derived automatically from a seller's intention.

Sub-supplier, sub-seller

A *sub-seller* represents an entry in a price-table of the seller. It identifies the seller's preference for selling one or more items with specific quantities for a specific price.

A sub-seller is represented by two data-structures: (1) An *RFQ* containing the requirements and ranges for quantities of items. (2) A relevant *GP model*, which contains constraints and preferences upon the items. The GP model is actually a sophisticated price formula for calculating the price of a package, instead of using a simple price-per-unit equation.

Package

The DS algorithm operates on *packages*. A package is an explicit deal, generated from a sub-seller. A package specifies exact quantities of the items to be purchased with a total price for the purchase.

Final deal

The *final deal* is a collection of packages with their relevant total prices. The final deal's *total price* is the sum of these prices.

General Item (GI):

This is a virtual item. It contains a collection of real items that *must* be supplied by the *same* supplier. Notice that it is the buyer who defines GIs. A supplier that knows the buyer's published RFQ may define its sub-suppliers while taking into account the buyer's GIs, however, this should not impose any preference to use such sub-suppliers in the system's solution.

Item ID.

An item is identified via a notion (i.e., type, e.g., Mouse, KB, CPU) and a name. A name is usually a serial identification number, which is usually used to distinguish simple items that should be provided within a GI and those that should be provided as "*stand alone*" simple items.

RFQ

Request For Quote is a general term describing a buyer's request for purchasing specific items with specific quantities (or ranges of quantities). See example below in the description of the input.

RRFQ or RRF

Response to RFQ is a general term describing a seller's offer for selling specific items of specific quantities (or ranges of quantities). An RRF also specifies a price-per-unit for each of the items provided, or a general function for calculating a total-price of a package from the RRF.

Input

Buyer *(A single buyer)*

1. RFQ (buyer's requirement)

An RFQ of Simple Items and General Items with their relevant attributes

Example

GI Flag	***Item Notion***	***Item Name***	***Quantity***
F	Mouse	0	30-80

- *Notice* that this is the exact requirement of the buyer, so that the suppliers can respond according to this explicit request by the buyer.

GP model

The buyer preferences and constraints are represented via a GP model. Notice that the model may be very complex. *It may impose constraints upon* the attributes of each item, such as on delivery date, color, quality, etc. as well as trade-offs between these attributes. Moreover, the GP contains objectives, describing penalties or bonuses upon deviating from the specified goal target(s).

The model is used to evaluate the packages of the suppliers in terms of the buyer's currency (that is, his GP), via a special utility that evaluates the packages even when a certain package does not fulfill the whole requirement of the buyer (according to the "quantity additive deals" assumption; *see Stage III. Evaluating the* packages.)

Supplier *(Multiple sub-suppliers)*

2. Availability vector

This is a vector describing the available quantities for the items that the supplier provides. General Items are not specified here, as only the buyer defines them.

Example

Item	*Tire*	Body	Radio	Car
Quantity	200	10	50	na

Sub-suppliers

A sub-supplier represents a collection of items that must be bought together as a whole. Each item in the package has specific attributes and ranges of values for the attributes.

A sub-supplier is represented, similarly to a buyer, via an RRF (Response to RFQ) and a GP model.

Algorithm Flow

Stage I. "Splitting" the buyer

The input RFQ of the buyer may include ranges of desired quantities, of the various items, for purchase. However, the DS system used is based upon exact quantities. That is, given a vector *need* of the required quantities for each item, a DS algorithm seeks a solution such that the purchase includes at least *need* quantities for every item, with minimum overall price. Moreover, the *need* values express minimum required quantities, and the DS algorithm cannot handle maximum values, and therefore cannot handle ranges in the *need* vector.

a. Input:

- *Buyer RFQ, quantities and GI clearly marked*
- *K, number of maximum specified deals allowed,*

Output

- *Up to K different derived "new" RFQs for the Buyer, where each RFQ contains explicit values (non-ranges) for the quantity variables of all the items (i.e., a need vector expressing quantities in terms of (simple) items and GIs).*

Algorithm outline

The basic idea is to choose the most important R items (the user should define the importance of the items); and allow quantity combinations only for these items, when the other items will have a randomly chosen quantity value within their range. Alternatively, we will allow users to specify desired quantities for items and use these targets for the less important items (these targets need not affect the GP's objective functions).

Example: Suppose we have five items with the ranges below, when *I2* and *I4* are the important items, and we would like to have at most K=8 combinations:

I1: [1-10], *I2*: [20-70], *I3*: [1-100], *I4*: [50-70], and *I5*: [2-6]

Then, for instance, *I2* and *I4* will have the representative combinations:

[30,50], [30,57], [30,63], [30,70], [50,50], [50,57], [50,63], [50,70]

For the other items, we'll choose at random, within their ranges, quantity values for each combination. Alternatively, if target quantities were specified for the other items, these will be used instead of a random selection.

Stage II. Preparing Sellers' Packages

The MRG algorithm (MI-DS greedy algorithm) works upon explicit packages of values for the attributes, where each package has a specific price value for purchasing it. As such, it considers only the quantity attribute, and prepares all the possible packages of a sub-supplier by generating all the possible combinations of unique quantity values for all items.

The algorithm described herein is used in order to generate all the possible packages for the MRG algorithm. However, due to the concept of *General Items,* further treatment is required, as introduced in Stage IV.

b. Input:

- *A list of Sellers, with a list of RFQs (sub-suppliers) for each seller.*
- *K, the number of maximum allowed packages per one seller. The default value, which is Infinity, allowing all the possible packages.*
- *S (optional), if K is a finite number, then the user must specify the most important items. Target quantities may be specified for the items.*
- *The buyer's RFQ* (indicating the GIs, if any).

Output

- *Up to K (or all the possible) quantity-based packages per seller.*

Algorithm outline

1. *If (K = Infinity) then* create all the basic packages.
2. *Else, choose the most important items (defined in S) and allow combinations only for these items, when the other items will have a randomly chosen quantity value within their ranges. See the algorithm outline of Stage I. "Splitting" the buyer. In case target quantities were specified for items not in S, they may be used instead of a random*

Discussion

It is recommended that a pre-processing check be done to verify that each of the buyer's items may be supplied by at least one seller.

Stage III. Evaluating the packages

Partial evaluation of GP objective function (*first level of GP only*)

c. Input:

- *Buyer's GP model.*
- *Buyer's Need Vector.*
- *A seller's GP model.*
- *A seller's offered package to evaluate.*

Output

- *The value of the buyer's objective function first level.*

Algorithm outline

1. Build a new GP model (GP_{eval}):
Merge the seller's GP and the buyer's one, and set the objective of the new GP to be the first level of the buyer's objective.

2. Set values for the *GI* variables:
See discussion below.

3. Set values for the *quantity* variables:
For every quantity variable in the Need vector, set the relevant quantity value (add a hard-bound constraint to GP_{eval}).

4. Adjust the objective function for *all* the variables:

a. For all the variables that are in the RFQ, multiply the relevant deviation variables (in the objective function) with the *fraction quantity*, which is the offering package quantity value divided by the Buyer Need Vector quantity value (the intuition here is discussed subsequently).

b. For missing variables in the offering package, deviation variables are set to zero.

5. Solve the GP_{eval} model and return the value of its single level objective function.

Discussion

1. Note the following concerning the evaluation of a package:

- The value of a package (price) is evaluated in the buyer's currency.
- Still, a package is at all valid, if it conforms to the seller's constraints (as expressed in the sub-seller's GP).
- Therefore, the calculation of price can be thought of as that of a "knowledgeable" sub-seller who knows his own restrictions as well as the buyer's GP.

2. The evaluation of a package is done according to the estimated final- total-quantity of the deal, as it appears in the Buyer's Need Vector. However, the DS algorithm may solve the problem with *higher* quantity values than those that appear in the Need vector, and in this case the value we calculate here is not accurate and is therefore an approximation.

Stage IV. Preparing GI Packages

d. Input:

- *Packages with their prices as computed in Stage III.*

Output

- *Additional packages containing GIs.*

Algorithm outline

For every basic package in the input:

1. For all the GIs in the Buyer's RFQ, $g_1, \ldots, g_h$
   - i. And *for every* basic package generated in stage II
     - a. *Generate all* the possible combined multiplicities of $g_1, \ldots, g_h$ *whose overall item total is* within the package, leaving the residue values in the original items.
     - b. Keep the basic package's price with the new packages thus generated.

Discussion

We can enhance the support for GIs by trying to combine more than one sub-seller of the same seller. We may run the MRDS algorithm upon a specific seller, and generate semi-deals representing combined sub-sellers that can provide the GIs.

Stage V. Execute MRG

This stage is concerned with running the MRG algorithm, after preparing its input, to find an approximate solution to the best combination of packages (with minimum price).

3. MRG

e. Input

- *A buyer's Need Vector.*
- *A collection of quantity-based packages and their explicit prices from all sellers. These are the packages prepared in stages II and IV with their prices calculated in stage III.*

Output

- *A Deal structure:* A list of packages specifying a deal composed of sub-suppliers together with an identification of their original suppliers.

Algorithm outline

See the section describing MRG.

Note (MRG implementation issue): The Need Vector is already in terms of GIs and the MRG will work transparently using MRG-alpha values accordingly.

Stage VI. Final Deal Price for a "split-buyer"

Re-calculate the price of each package in the final deal, but this time instead of using the Need Vector uses the *actual* quantity values as calculated by MRG.

The final price is the sum of prices of the deal's packages.

Note: It might be instructive to display the difference between the two sums of the packages' prices (according to the Need Vector, and according to the MRG result) so as to hint at the accuracy of the approximation.

Stage VII. Final Overall Deal Price for the Buyer

Among all the deals of split buyers calculated in stage VI, obtain the best one for the buyer.

Reference is now made to Fig. 50, which is a simplified diagram of a process of deal splitting.

Discussion

A deal is made of several packages. We need to explain how to assign a value, or a score, to a deal. One desirable property is that rearranging the same package in a different way, as a deal split into sub-packages will not result in a different value. For this, we define the notion of a *consistent deal composition scheme*.

We would also like to combine different packages in a deal and weight each package's contribution according to the quantities of items it has. Naturally, the question arises as to whether this operation is the "right thing to do". We show, however, that if our package evaluation function enjoys a certain desirable property, i.e., *quantity-additive* or *good*, then this relative weighting composition defines a *consistent composition.*

This lends credibility and legitimacy to our weighting scheme.

Our argument is expressed below through a series of definitions and claims:

Deal Composition functions (for Deal Splitting evaluation)

If a buyer's RFQ *were to be satisfied by one package,* then we can calculate the exact cost of the package in terms of the buyer's "currency". However, as we are using deal-splitting techniques, we'll usually need many packages (offers from different sub-sellers) to cover the buyer's request. Below we explain how and when is it possible to split a deal and calculate its price by combining the prices of its packages. Suppose that:

- The *deal* is a collection of packages $P_1, \ldots, P_m$.

- Each package P_i is a collection of *items* $I_1, \ldots, I_k$.
- In package P_i, each item I_j is associated with *values* $a_{i,j,1}, \ldots, a_{i,j,at(j)}$ for *attribute* set $A_{j1}, \ldots, A_{at(j)}$. Without loss of generality, for all j: A_{j1} is the *quantity* attribute for item j, and its values $a_{i,j,1}$ will be described using the symbol q_{ij}.
- $f(\mathit{package})$: The value of a *package* is the *sum* of the values of the individual items in the *package*. That is, the package evaluation function evaluates each item independently of the other items, and as such, changing the attribute values of one item may not alter the evaluation of any other item in the package.
- $F_f(\mathit{deal})$: The value of a *deal* is the sum of the values of the individual packages. As such, F_f is called a *deal composition scheme.*

Consistent deal composition schemes

Consider a deal $D = P_1, \ldots, P_m$ where for each item *j* each attribute A_{ji}, such that $i > 1$ (i.e., the non-quantity attributes), has the same value (e.g., for item 5, all the date attributes have the same value).

Also consider a single package deal, whose only package is P_D, which for each item *j*, has the same attribute values as in D for the non-quantity attributes, and whose quantity attributes are $q_{Dj} = \sum_{i=1,,m} q_{ij}$. That is, all the items of certain kind are coalesced.

We define a deal composition scheme to be *consistent*, when:

$$f(P_D) = F_f(D).$$

Intuitively, this means that taking a package and breaking it quantity-wise while keeping the other attributes unchanged, results in a deal with the same value as that of the pre-split package.

Reasonable package evaluation functions

First we define the following properties over functions for evaluating packages. Recall that a package evaluation function is the sum of individual items in the package.

A *quantity independent function* over packages, f_α, is such that given a package P_i, then for each item *t*, given quantities $q_{ij}=0,\ j\neq t$

$$f_\alpha(0,\ldots,a_{i,1,at(1)},\cdots,q_{it},\ldots,a_{i,t,at(t)},\cdots,0,\ldots,a_{i,k,at(k)}) = \\ f_\alpha(0,\ldots,a_{i,1,at(1)},\cdots,1,\ldots,a_{i,t,at(t)},\cdots,0,\ldots,a_{i,k,at(k)})$$

, f_α, and a quantity independent function $D = P_1, \ldots, P_m$ Given a deal

, such that given a $f_{\alpha^\bullet}$ we define a *quantity additive function* over packages,

, $q_{ij}=0,\ j\neq t$ then for each item *t*, given quantities P_i package

denotes the argument in $1.\bullet$ where $f_{\alpha^\bullet}(\bullet) = \left(q_{ij}/Q_j\right)\cdot f_\alpha(\bullet)$

Notice that $Q_j = \sum_{i=1..m} q_{ij}$, i.e., the total quantity of item *j* in all the packages of *D*.

, is such that given a package f_β A *consistent function* over packages,

P_i

$q_{ij}=0,\ j\neq t$ then for each item *t*, given quantities

$$f_\beta(0,\ldots,a_{i,1,at(1)},\cdots,q_{it},\ldots,a_{i,t,at(t)},\cdots,0,\ldots,a_{i,k,at(k)}) = \\ q_{it}\cdot f_\beta(0,\ldots,a_{i,1,at(1)},\cdots,1,\ldots,a_{i,t,at(t)},\cdots,0,\ldots,a_{i,k,at(k)})$$

q_{it}, is q_{it} That is, the value of a package consisting of one item with quantity times the value of the same package with only one quantity purchased.

, f_β, and a consistent function $f_{\alpha^\bullet}$ Given a quantity additive function

we define a *good function* over packages, as follows:

$$f_{good}(p) = f_{\alpha^\bullet}(p) + f_\beta(p)$$

Claim. For every quantity additive function f, the scheme F_j is consistent.

Claim. For every good function f, the scheme F_f is consistent.

Note. The deal-splitting algorithms described in the DS section evaluate packages by requiring a *price-per-unit* specification per each item. However, in this section we are presenting an approach where a package is evaluated as a whole via a general mathematical method, such as GP models. Therefore, we ask for a weaker relationship between the items prices and the package's price in the form of the quantity-additive property.

It is appreciated that certain features of the invention, which are, for clarity, described in the context of separate embodiments, may also be provided in combination in a single embodiment. Conversely, various features of the invention which are, for brevity, described in the context of a single embodiment, may also be provided separately or in any suitable subcombination.

It will be appreciated by persons skilled in the art that the present invention is not limited to what has been particularly shown and described hereinabove. Rather the scope of the present invention is defined by the appended claims and includes both combinations and subcombinations of the various features described hereinabove as well as variations and modifications thereof which would occur to persons skilled in the art upon reading the foregoing description.

Claims

1.A platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit for:

defining respective party's goal programs in respect of said outcome, said goal program comprising a plurality of objective functions and constraints associated with respective objective functions,

for associating each of said objective functions with one of a plurality of levels of importance, and

for assigning to objective functions within each level a respective importance weighting, said party goal program unit comprising a party input unit for allowing a party to provide data for a respective goal program,

a goal program unifier, associated with said party goal program unit for receiving goal programs of respective parties, and carrying out unification of said goal programs by considering said objective functions objectivewise and levelwise with associated constraints in the respective goal programs to determine whether two goal programs have a common field of interest from which a mutually compatible outcome is derivable,

a negotiator associated with said goal program unifier for receiving goal programs of respective parties, and carrying out negotiations using said goal programs by considering said objective functions objectivewise and levelwise with associated constraints in the respective goal programs to arrive at said mutually compatible outcome by carrying out minimization firstly objectivewise and then levelwise, therewith to form an offer, an output unit for offering said unified goal program to said respective parties,

a response receiver for receiving from respective parties either counter offers or acceptances, said response receiver being operable to provide counter offers as new goal programs to said goal program negotiator for further unification.

2.The platform of claim 1, wherein said negotiator comprises a trade-off unit for identifying excludable objectives levelwise in a first party's goal program for conditional weakening from said outcome in a trade-off involving strengthening of other objectives within the same level of said first party.

3.The platform of claim 1, wherein said negotiator comprises a trade-off unit for identifying excludable objectives levelwise in a first party's goal program for conditional weakening from said outcome in a trade-off involving weakening of other objectives within the same level of another party.

4. The platform of claim 1, wherein said party goal program unit is operable to express said objective functions in a weighted hierarchy according to the respective associated level of importance, and to express each constraint in terms of a constraint variable, thereby to form an expression for minimization at said negotiator.

5.The platform of claim 4, wherein said party goal program unit is operable to use data from said party data input unit to apply coefficients to said constraint variables.

6.The platform of claim 5, wherein said party goal program unit is operable to apply user input to provide different values of coefficients to said constraint variables for deviations of a corresponding objective in respective positive and negative directions.

7.The platform of claim 6, wherein said objective program unit is operable to apply said user input to apply said coefficient values to define any one of a group comprising:

a strong one sided objective,

a weak one sided objective,

a complex single sided objective,

a simple two sided objective,

a complex two sided objective,

a simple first side-complex second side objective,

a simple two-sided objective with an indifferent range,

a complex two sided objective with an indifferent range, and

a simple first side-complex second side simple objective with an indifferent range.

8.The platform of claim 1, wherein at least one objective comprises a series of discrete values.

9.The platform of claim 8, wherein said party goal program unit is operable to apply user input to formulate weightings for respective ones of said discrete values, thereby to express a preference between said discrete values.

10.The platform of claim 4, wherein at least one objective function comprises a continuous variable, and wherein said party goal program unit is operable to apply user input to determine whether said continuous variable is to be maximized or to be minimized.

11.The platform of claim 4, wherein said negotiator is operable to arrange said levels in a hierarchy and to carry out minimization by summing objective functions associated with constraint variables levelwise in said hierarchy.

12.The platform of claim 4, wherein said negotiator is operable to carry out minimization by carrying out minimization per constraint variable and setting a maximum bound per deviation.

13.The platform of claim 4, wherein said negotiator is operable to carry out minimization for an expression comprising first ones of said constraint variables and then to add further constraint variables to said expression for a further minimization stage.

14.The platform of claim 1, wherein said party input unit is configured to receive data from a user interface.

15. The platform of claim 1, wherein said party input unit is configured to receive data from a software agent.

16. The platform of claim 1, wherein said party input unit is operable to identify parameter data missing from an input and further comprises a default value generator for generating said missing parameter.

17. The platform of claim 1, wherein said party input unit is operable to identify parameter data missing from an input and further comprises a default register of values for expected parameters.

18.The platform of claim 1, wherein said party input unit is operable to obtain lower and upper bounds for at least some of said objective functions.

19.The platform of claim 18, wherein goal program unit is operable to use said upper and lower bounds to express deviations of a respective objective function from a target value relatively, thereby to render said deviations subject to comparison by said unifier or by said negotiator.

20.The platform of claim 1, wherein said party input unit is operable to obtain a objective function interval, and a value for a penalty for deviating from said interval, and wherein said unifier is operable to define a working interval between two objective functions as an intersection between respective intervals.

21.The platform of claim 20, wherein said unifier is operable to determine that a target value of one of said objective functions is outside said working interval, and to modify said target value to approach a closest boundary of said working interval.

22. The platform of claim 21, wherein said unifier is operable to apportion said penalty in accordance with said target value modification.

23. The platform of claim 22, wherein said intersection is a point.

24. The platform of claim 22, wherein said party goal program unit comprises operability for determining that an intersection is small to the satisfaction of respective parties and, when said intersection is recognized as small, said negotiator is operable to select a point within said intersection being a midpoint between respective target values.

25. The platform of claim 19, wherein said negotiator is operable to measure deviations within said interval as a fraction of a total size of said interval.

26. The platform of claim 25, wherein said party goal program unit is operable to obtain importance values for deviations from said target and wherein said negotiator is operable to use said importance value as a multiplier to measure said deviation.

27. The platform of claim 20, wherein said negotiator is operable to identify intersections that are small and distant from a target value compared to one of said objective functions and large and inclusive of a target value compared to another of said objective functions, to set an effective target at the closest intersection boundary and to set a transformed deviation as giving the arithmetic deviation when multiplied

by the effective target and then added to the difference between the old target and the effective target, to produce a result which is divided by the old target.

28. The platform of claim 1, wherein said party input unit is operable to permit a party to define at least one single dimension interval objective in respect of said outcome, and to associate said objective with a range of indifference having an upper bound and a lower bound, a first weighting value for deviations below said lower bound, a second weighting value for deviations above said upper bound and a relative importance for said objective,

said unifier being operable to use said range of indifference, said weightings and said relative importance to unify said at least one objective with at least one other objectives to determine said compatibility.

29. The platform of claim 28, wherein said at least one other objective is a corresponding objective in a goal program of an opponent.

30. The platform of claim 28, wherein said party input unit further comprises a prioritizer for allowing said respective party to define said association of a respective objective function levelwise with other objectives, said party input unit further being operable to allow said party to establish a priority with objectives within a same level.

31. The platform of claim 1, wherein said party input unit is operable to permit a party to define a two dimensional trade-off objective by entering two two-dimensional points, said party goal program unit being operable to define a trade-off line between said two points.

32. The platform of claim 31, wherein said party input unit is operable to permit a party to define weights for deviation from said trade-off line.

33. The platform of claim 31, said party input unit being operable to permit a party to define a relative importance for said two dimensional trade-off objective.

34. The platform of claim 33, wherein said party input unit is operable to permit a party to associate said two dimensional trade-off objective levelwise with other objectives.

35. The platform of claim 34, wherein party enterable weightings are usable by said unifier to establish a priority between objectives in a same level.

36. The platform of claim 1, wherein said party input unit is operable to allow a party to define at least one single dimension two-point objective in respect of said outcome, and to associate said objective with an upper point of preference, and a lower point of preference, a first weighting value for deviations below said lower point of preference, and a second weighting value for deviations above said upper point of preference, said goal program unit being operable to provide weightings to a region included between said points of preference by assigning said first weighting value below said upper point of preference and said second weighting value above said lower point of preference and defining an overall weighting within said region as a minimum of said weighting values,

and wherein said unifier is operable to use said range of indifference, said weightings, and said minimum to unify said at least one objective with other objectives to determine said compatibility.

37. The platform of claim 36, wherein said party input unit is operable to permit a party to define a relative importance for said single dimensional two point objective.

38. The platform of claim 37, wherein said party input unit is operable to permit a party to associate said single dimensional two point objective levelwise with other objectives.

39. The platform of claim 38, wherein said relative importance is usable by said negotiator to establish a priority between objectives in a same level.

40. The platform of claim 1, wherein said party input unit is operable to permit a user to define a piecewise linear two-dimensional goal by entering at least three two-dimensional points, said party goal program unit being operable to define a trade-off line between said three points,

said negotiator being operable to apply penalty values to points on said trade-off line in accordance with their distances from said points.

41. The platform of claim 40, wherein said party input unit is operable to permit a user to define a first deviation weight for deviating in a first direction from

said trade-off line and a second deviation weight for deviating in a second direction therefrom.

42. The platform of claim 1, wherein said party input unit is operable to permit parties to define objectives comprising pairwise trade-offs having at least two points and a trade-off therebetween, and to associate each of said trade-offs with a slope.

43. The platform of claim 42, wherein said party goal program unit is operable to prevent inconsistent trade-offs to be defined within the platform by preventing said party input unit from accepting more than one trade-off from referring to any given point.

44. The platform of claim 42, wherein said party goal program unit is operable to warn users of inconsistent trade-offs by outputting a warning whenever a trade-off being entered has a point already included in a previously entered trade-off.

45. A platform according to claim 1, wherein said party input unit is further operable to allow a party to define disjunctive constraints in respect of objectives, said goal program unit comprising a disjunctive constraint processor for translating a disjunctive expression into at least one linear conjunctive expression, and wherein said unifier is operable to utilize said linear conjunctive expression to unify said at least one objective with other objectives to determine said compatibility.

46. The platform of claim 45, wherein said disjunctive expression comprises a series of relationships including equality relationships.

47. The platform of claim 46, wherein said disjunctive constraint processor is operable to carry out said translation by expressing at least one of said equality relationships as the union of two corresponding inequalities that meet at a point of equality of said equality relationship.

48. The platform of claim 46, wherein said disjunctive constraint processor is operable to define binary variables for said relationships, for setting wherever said relationships are satisfied, and wherein said negotiator is operable to sum said variables to determine a satisfaction level for said objective.

49. The platform of claim 48, wherein said party goal program unit is operable to set a requirement of a minimum number of satisfied relationships for use by said negotiator.

50. The platform of claim 1, wherein:

said party input unit is further operable to permit each party to define weighting values for a discrete variable predefined per outcome, for use in said goal program definition, and

said negotiator being operable to carry out negotiation of said goal programs by considering said weighting values to arrive at an outcome comprising an offered one of said values.

51. The platform of claim 50, wherein said party goal program unit is operable to use said weightings for respective values of said discrete variable to arrange said

variable as a weighted hierarchy, said hierarchy being usable by said negotiator to arrive at said offered one of said discrete values.

52. The platform of claim 51, wherein said party goal program unit is operable to use said values and respective weightings to build summation functions, therefrom to express said variable in quantitative manner.

53. The platform of claim 52, wherein said negotiator is operable to attempt offer formation by setting one of said discrete values to one and the remainder to zero, and then to calculate said summation functions to contribute to a fulfillment level of each goal.

54. The platform of claim 54, wherein said negotiator is operable to reattempt offer formation by setting different ones of said discrete values to one, thereby to find a value which maximizes said fulfillment level.

55. The platform of claim 1, wherein said party goal program unit is operable to represent date information as accumulated minutes from a threshold starting date, and further to modify said dates relative to upper and lower bounds entered via said party input unit.

56. A platform according to claim 1, wherein said party input unit is further operable to allow input of variables in association with said objective functions and a linkage between a first and a second of said variables, said linkage defining a trade-off path of deviations with respect to said target values, said negotiator being operable

to use said series of variables including said trade-off path to negotiate an outcome in respect of said at least one objective with other objectives, thereby to arrive at formation of an offer.

57. The platform of claim 56, wherein said party goal program unit is operable to express said second variable as a function of said first variable, thereby to represent said linkage, and wherein said negotiator is operable to represent deviations from respective target values as deviations from the target value of said first variable.

58. The platform of claim 56, wherein said party goal program unit is operable to express said trade-off as separate deviation variables in respect of said first variable and in respect of said second variable wherein said separate deviation variables are orthogonal.

59. The platform of claim 56, wherein said party input unit is operable to permit an association of a relative importance level to said trade-off.

60. The platform of claim 56, wherein said party goal program unit is operable to calculate a relative importance level for said trade-off as an average of respective relative importance levels of said first and second variables.

61. The platform of claim 1, wherein said unifier comprises a goal program generalizer to form a generalization of received goal programs for use in said unification.

62. The platform of claim 1, wherein said party goal program unit is operable to translate said input received from said party input unit into said objective functions and constraints on said objective functions within said goal program, said negotiator comprising an optimizer to find best values for said objective functions and constraints, therewith to obtain a best solution for the goal program for output as a first offer, and then iteratively to produce further solutions until an offer is accepted, thereby to achieve said outcome.

63. The platform of claim 62, wherein said negotiator comprises a percentage reducer for taking ones of said objective functions in turn, worsening them by a predetermined percentage, thereby to produce said further solutions.

64. The platform of claim 63, wherein said percentage reducer is settable to take each of said objective functions in turn beginning with a most important objective, until a solution is accepted.

65. The platform of claim 63, wherein said objective functions are arranged in levels and said percentage reducer is settable to take objective functions of a first of said levels only.

66. The platform of claim 62, wherein said negotiator comprises a worst case calculator for determining a worst case level for ones of said objective functions and constraints, thereby to obtain a worst acceptable offer.

67. The platform of claim 62, wherein said goal program is arranged as pairwise bounded variables, and said negotiator comprises an arbitrary case calculator for taking one of each pair of variables, setting it to an arbitrary value within its respective bounds, taking the other of said pair of variables and setting it to zero, therefrom to calculate ones of said iterative solutions.

68. The platform of claim 66,wherein said negotiator further comprises an average case calculator, associated with said optimizer and with said worst case calculator, for

taking said best solution and said worst solution, each solution having corresponding values,

associating ones of said corresponding values, and

constraining variables of said goal program towards an average of each of said corresponding values, therewith to provide an average solution.

69. The platform of claim 68, wherein said goal program objective functions are arranged levelwise and said average case calculator is operable to carry out said associating and said constraining successively levelwise, thereby to produce said series of iterative solutions.

70. The platform of claim 1, wherein said negotiator comprises a solution sorter for comparing goal program solutions by solving said goal program for each one of a series of solutions and ranking the solutions, said negotiator being operable to use said ranking to apply preference to different solutions.

71. The platform of claim 70, wherein said negotiator further comprises a thresholder associated with said solution sorter for applying a threshold to said evaluations to exclude ones of said series of solutions.

72. The platform of claim 71, wherein said solution sorter further comprises a solution completer for applying best values to incompleted variables in incomplete ones of said solutions, thereby to allow said goal program to be evaluated for said incomplete solutions.

73. The platform of claim 70, wherein said solution sorter is settable to find the best solution from said series of solutions by identifying the highest ranked solution and discarding the remaining solutions.

74.The platform of claim 70, wherein said solution sorter comprises a memory, set to hold a predetermined number of solutions, and a comparator to compare a new solution with each solution in said memory, and further comprising a control unit for adding said new solution to said memory if its evaluation is larger than any solution in said memory, and for discarding a lowest ranked solution in said memory.

75.The platform of claim 1, wherein said goal program unit comprises a data input unit for receiving user defined output values, and wherein said goal program unit is operable to set said output values as single value constraints and to flag said constraints as unchangeable for said unifier.

76.The platform of claim 75, wherein said data input unit is operable to output an error indicator if said single value constraints render said goal program insoluble.

77.The platform of claim 1, wherein the unifier further comprises a goal program input unit for receiving a local party's goal program and an opponent's goal program to be unified therewith, said goal programs comprising objective functions associated with constraints and being arranged in levels, and the negotiator further comprises:

an optimizer for finding best solutions to goal programs, connected to find best values for said objective functions and constraints of said local party's goal program levelwise, and

a worst case calculator for finding worst solutions for goal programs, connected to find worst values for said objective functions and constraints of said opponent's goal program levelwise,

said negotiator being operable to:

use said optimizer and said worst case calculator in succession level by level to produce successive value sets for evaluation therefrom to form level by level unification offers, and

advance from one level to another level only following acceptance by said parties of a unification offer regarding a previous level.

78.The platform of claim 77, wherein said negotiator comprises a constraint updater for updating constraints upon advance from one level to another level in accordance with said respective acceptance.

79.The platform of claim 77, wherein said negotiator further comprises an offer improver operable to provide an improved offer by making a change in a selected one of said variables to bring about an improvement in the evaluation of the opponent's goal program.

80.The platform of claim 79, wherein said change in said variable is calculated such that said improvement is a predetermined proportion of a difference between a previous offer made and a best possible evaluation made for the opponent.

81.The platform of claim 80, wherein said offer improver is operable to use a value of said selected variable in a last opponent's offer to moderate said change.

82.The platform of claim 80, wherein said offer improver is operable to calculate a protection value, and to use said protection value to limit a reduction in the evaluation of the local party's goal program as a consequence of said improvement to the opponent's goal program evaluation.

83.The platform of claim 82, wherein said protection value is a proportion of the difference between a worst case evaluation of the local party's goal program and an evaluation of last previous offer thereof.

84.The platform of claim 77, further comprising an offer improver for taking goal program values of a previous local party offer and one value in turn from a previous opponent offer, testing the opponent value against local constraints, and if it fits within the constraints then substituting it into the previous local party offer thereby to provide an improved offer.

85.The platform of claim 1, wherein the negotiator further comprises a goal program input unit for receiving a local party's goal program, said goal programs comprising objective functions associated with constraints and being arranged in levels, and said negotiator further comprises:

an optimizer for finding best solutions to goal programs, connected to find best values for said objective functions and constraints of said local party's goal program levelwise, and

a stay close processor for determining variable improvement directions from monitoring of received offers from said opponent and carrying out value perturbations in said directions,

said negotiator being operable to:

use said optimizer to produce a first offer for a first level,

to advance from one level to another level only following acceptance by said parties of an offer regarding a previous level, and

use said stay close processor to produce a first offer for each subsequent level, thereby to arrive at said outcome.

86.The platform of claim 85, wherein said negotiator comprises a constraint updater for updating constraints upon advance from one level to another level in accordance with said respective acceptance.

87.The platform of claim 85, wherein said negotiator further comprises a gap value determiner, for determining a gap for use in offer improvement, and

a value improver, associated with said gap value determiner, for inserting a predetermined proportion of said gap as a constraint of said goal program, and wherein said stay close processor is associated with said value improver thereby to apply predetermined gap proportion in said direction to provide an improved offer.

88.The platform of claim 87, wherein said stay close processor is operable to monitor two successive opponent offers for value changes therebetween, and to assign to each respective changing variable a weight for use in providing an improved offer, the magnitude of said weight being selected in accordance with a monitored relative size of a corresponding value change of said opponent.

89.The platform of claim 87, wherein said gap is a constant.

90.The platform of claim 89, wherein said constant is a difference between a best value and a worst value of a corresponding variable.

91.The platform of claim 87, wherein said gap is a difference between a last local proposal and a last opponent proposal.

92.The platform of claim 1, further comprising a negotiation necessity tester, associated with said unifier, for joint solving of said local and said other goal program to form a joint goal program comprising optimal solutions for each of said local and said other goal program, said negotiation necessity tester being set to determine whether there lies a single solution that includes both optimal solutions within said common ground, and if so, to inhibit passing of said goal programs to said negotiator.

93. The platform of claim 1, further comprising a mediation unit callable by both parties levelwise during said negotiations, said mediation unit being operable to retain agreed objectives of previously solved levels and to apply a summation formula to solve a current level.

94. The platform of claim 1, wherein said summation formula is:

$$\left\{\frac{\sum w_k^+ \cdot \delta_k^+ + \sum w_k^- \cdot \delta_k^-}{\sum w_k^+ + \sum w_k^-}\right\} + \left\{\frac{\sum v_k^+ \cdot \gamma_k^+ + \sum v_k^- \cdot \gamma_k^-}{\sum v_k^+ + \sum v_k^-}\right\}$$

wherein k is a number of a current level, objective, + and – represent respective sides of a target value, v is a , and γ is a

95. The platform of claim 1, further comprising a mediation unit callable by both parties during said negotiations, said mediation unit being operable to stop operation of said negotiator, apply a summation formula to provide a median solution between respective goal programs, and to provide said median solution as an offer to both parties.

96. The platform of claim 94, wherein each goal program is expressed as a series of decision variables each having an upper bound, a lower bound, a target value and one or more constraints, the platform further comprising a form offer unit for providing a form offer to the parties, the unit being operable to assign to each of the goal programs a weighting such that the sum of the weightings is unity, for each

variable and to calculate the offer by minimizing relative deviations for each variable over said goal programs weighted according to said assigned weightings.

97.The platform of claim 1, further comprising a discrete variable form offer unit operable to transform values of said discrete variable into a continuous domain, to carry out minimization in light of goal program objectives of said two parties in said continuous domain, and to transform the minimization results back to discrete values, thereby to provide a form offer.

98.The platform of claim 1, wherein said negotiator is operable to provide a value for at least one objective by comparing said at least one objective from a local goal program against the entirety of an opponent goal program.

99.The platform of claim 1, further comprising an item catalog for storing a plurality of items for offer in terms of values of said objectives, wherein said negotiator is operable to provide offers in terms of nearest items in said catalog.

100.The platform of claim 99, wherein said negotiator comprises:

an item manager for recursively determining which items of said catalog are currently within the scope of negotiations,

a first round manager, associated with said item manager, for managing levelwise goal program negotiation to successively reduce the number of said items within said scope to a predetermined threshold number of items,

a second round manager, associated with said item manager, for managing levelwise program negotiation to produce successive offers, and an item associator, connected to said second round manager and to said item manager, for expressing said successive offers in terms of items within said scope.

101.The platform of claim 100, wherein said item manager is operable to measure distance from said scope in terms of an opponent goal program.

102.The platform of claim 100, wherein said item manager is operable to measure distance from said scope in terms of a joint goal program.

103.The platform of claim 100, wherein said item manager is operable to measure distance initially in terms of a local goal program, to order said items and iteratively to remove most distant items.

104.The platform of claim 99, wherein one of said objectives is compatibility with a second item.

105.The platform of claim 1, further comprising an offer delay timer, associated with said negotiator, for introducing determinable delays between issuance of successive offers to an opponent.

106.The platform of claim 105, wherein said offer delay timer is operable to set successively increasing delays.

107.The platform of claim 106, wherein said offer delay time is operable to set delays to change levelwise.

108.The platform of claim 105, wherein a magnitude of said delay is based on a relative change between succeeding opponent offers.

109.The platform of claim 105, wherein said offer delay timer is operable to set user determined delays.

110.The platform of claim 1, wherein at least one of said objectives comprises a dynamic variable.

111.The platform of claim 1, wherein at least some of said constraints are associated with dynamically changing variables.

112.A platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit for allowing each party to define a plurality of goals in respect of said outcome, and to associate each of said goals with a respective level of importance, therefrom to form for each party a goal program,

said party input unit being operable to obtain a target value and upper and lower bounds for at least one of said goals, said party goal program unit being operable to use said upper and lower bounds to express deviations from said target values in relative terms, thereby to render deviations from different goals comparable.

113.The platform of claim 112, further comprising a negotiator, operable to define an interval between said upper bound and said lower bound and to measure deviations within said interval as a fraction of a total size of said interval.

114.A platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit for allowing a party to define a plurality of goals in respect of said outcome, and to associate at least one of said goals with a target value, an acceptable interval, and a penalty for deviation from said interval, and

a unifier, for determining common ground between said goal program and at least one_other goal program,

and a negotiator, operable to form offers within said common ground by mutual quantifying of a objective function of said at least one goal program with objective function of said at least one other goal program having a target value and an interval, by determining an intersection between said intervals and if said target value is outside said intersection then moving said target value by a deviation amount to a closest boundary of said intersection, said negotiator further being operable to apportion said penalty for deviation amount in accordance with an extent of said deviation of said target value.

115.The platform of claim 114,wherein said intersection is a point.

116.The platform of claim 114, wherein said party goal program unit comprises operability for determining that an intersection is small to the satisfaction of respective parties and, when said intersection is recognized as small, said unifier is operable to select a point within said intersection being a midpoint between respective target values.

117.The platform of claim 114, wherein said negotiator is operable to measure deviations within said interval as a fraction of a total size of said interval.

118.The platform of claim 117, wherein said party goal program unit is operable to obtain importance values for deviations from said target and wherein said negotiator is operable to use said importance value as a multiplier to measure said deviation.

119.The platform of claim 114, wherein said negotiator is operable to:

recognize intersections that are small and distant from the target value of said at least one goal and at the same time large and inclusive of a target value of said other goal,

set a unified target at the intervening intersection boundary at a determinable deviation from each goal,

calculate the deviation of said unified target from said distant target,

multiply said deviation by a value of the unified target,

add the result of said multiplication to the difference between values of the distant target and the unified target, and

divide the result by a value of the distant target, thereby to produce a transformed deviation value for said at least one goal.

120.The platform of claim 119, wherein said negotiator is further operable to normalize said deviation.

121.The platform of claim 119, wherein said negotiator is operable to normalize said transformed deviation.

122.The platform of claim 114, further comprising an offer delay timer, associated with said negotiator, for introducing determinable delays between issuance of successive offers to an opponent.

123.The platform of claim 122, wherein said offer delay timer is operable to set successively increasing delays.

124.The platform of claim 123, wherein said offer delay time is operable to set delays to change levelwise.

125.The platform of claim 122, wherein a magnitude of said delay is based on a relative change between succeeding opponent offers.

126.The platform of claim 122, wherein said offer delay timer is operable to set user determined delays.

127.A platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit for allowing a party to define at least one goal program having a plurality of objectives in respect of said outcome, and to associate at least one of said objectives with a range of indifference having an upper bound and a lower bound, a first weighting value for deviations below said lower bound, a second weighting value for deviations above said upper bound and a relative importance for said objective,

and a negotiator, associated with said goal program unit, said negotiator being operable to use said range of indifference, said weightings and said relative importance to obtain an outcome for said at least one objective in view of other objectives, by producing successive offers.

128.The platform of claim 127, wherein said party input unit further comprises a prioritizer for allowing said objective to be associated levelwise with other objectives.

129.The platform of claim 128, wherein said negotiator is operable to use said relative importance to establish a priority between objectives in a same level.

130.The platform of claim 127, further comprising an offer delay timer, associated with said negotiator, for introducing determinable delays between issuance of successive offers to an opponent.

131.The platform of claim 130, wherein said offer delay timer is operable to set successively increasing delays.

132.The platform of claim 131, wherein said offer delay time is operable to set delays to change levelwise.

133.The platform of claim 130, wherein a magnitude of said delay is based on a relative change between succeeding opponent offers.

134.The platform of claim 130, wherein said offer delay timer is operable to set user determined delays.

135.A platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit operable to permit a party to define a two dimensional trade-off objective by entering two two-dimensional points, said party goal program unit being operable to define a trade-off line between said two points, and

a negotiator, associated with said goal program unit, said negotiator being operable to use said trade-off line to unify said at least one objective with other objectives to arrive at said outcome via a series of successive offers.

136.The platform of claim 135, wherein said party input unit is operable to permit a party to define weights for deviation from said trade-off line.

137.The platform of claim 135, wherein said party input unit is operable to permit a party to define a relative importance for said two dimensional trade-off objective.

138.The platform of claim 137, wherein said party input unit is operable to permit a party to associate said two dimensional trade-off objective levelwise with other objective.

139.The platform of claim 138, wherein said relative importance is usable by said negotiator to establish a priority between objectives in a same level.

140.The platform of claim 135, wherein said negotiator further comprises a trade-off line evaluator for assigning a trade-off penalty value to points on said trade-off line.

141.The platform of claim 140, wherein said negotiator further comprises a scaling unit, associated with said trade-off line evaluator for scaling said trade-off penalty value, to produce a scaled penalty value.

142.The platform of claim 135, further comprising an offer delay timer, associated with said negotiator, for introducing determinable delays between issuance of successive offers to an opponent.

143.The platform of claim 142, wherein said offer delay timer is operable to set successively increasing delays.

144.The platform of claim 143, wherein said offer delay time is operable to set delays to change levelwise.

145.The platform of claim 142, wherein a magnitude of said delay is based on a relative change between succeeding opponent offers.

146.The platform of claim 142, wherein said offer delay timer is operable to set user determined delays.

147.A platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit for allowing a party to define at least one single dimension two-point objective in respect of said outcome, and to associate said objective with an upper point of preference, and a lower point of preference, a first weighting value for deviations below said lower point of preference, and a second weighting value for deviations above said upper point of preference, said goal program unit being operable to provide weightings to a region included between said points of preference by assigning said first weighting value below said upper point of preference and said second weighting value above said lower point of preference and defining an overall weighting within said region as a minimum of said weighting values,

and a negotiator, associated with said goal program unit, said negotiator being operable to use said range of indifference, said weightings, and said minimum to converge said at least one objective with other objectives to arrive at successive offers to achieve said outcome.

148.The platform of claim 147, wherein said party input unit is operable to permit a party to define a relative importance for said single dimensional two point objective.

149.The platform of claim 148, wherein said party input unit is operable to permit a party to associate said single dimensional two point objective levelwise with other objectives.

150.The platform of claim 149, wherein said relative importance is usable by said unifier to establish a priority between objectives in a same level.

151.The platform of claim 147, further comprising an offer delay timer, associated with said negotiator, for introducing determinable delays between issuance of successive offers to an opponent.

152.The platform of claim 151, wherein said offer delay timer is operable to set successively increasing delays.

153.The platform of claim 152, wherein said offer delay time is operable to set delays to change levelwise.

154.The platform of claim 151, wherein a magnitude of said delay is based on a relative change between succeeding opponent offers.

155.The platform of claim 151, wherein said offer delay timer is operable to set user determined delays.

156.A platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit operable to permit parties to define objectives comprising pairwise trade-offs having at least two points and a trade-off therebetween, and to associate each of said trade-offs with an inclination value, wherein said party goal program unit is operable to prevent inconsistent inclination values to be defined within the platform by preventing said party input unit from accepting more than one trade-off that refers to a same point

and a negotiator for negotiating with other parties via goal programs to achieve an outcome consistent with said objectives.

157.A platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit operable to permit parties to define objectives comprising pairwise trade-offs having at least two points and a trade-off therebetween, and to associate each of said trade-offs with an inclination value, wherein said party goal program unit is operable to warn users of inconsistent inclination values by outputting a warning whenever a trade-off being entered has a point already included in a previously entered trade-off, and

a negotiator for negotiating with other goal programs to achieve an outcome consistent with said objectives.

158.A platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit for allowing a party to define at least one objective in respect of said outcome, and to associate said objective with a series of variables including disjunctive constraints, said goal program unit comprising a disjunctive constraint processor for translating a disjunctive expression into at least one linear conjunctive expression,

and a negotiator, associated with said goal program unit, said negotiator being operable to use said series of variables including said linear conjunctive expression to negotiate an outcome consistent with said goal program and with other goal programs.

159.The platform of claim 158, wherein said disjunctive expression comprises a series of relationships including equality relationships.

160.The platform of claim 159, wherein said disjunctive constraint processor is operable to carry out said translation by expressing at least one of said equality relationships as the union of two corresponding inequalities that meet at a point of equality of said equality relationship.

161.The platform of claim 159, wherein said disjunctive constraint processor is operable to define binary variables for said relationships, for setting wherever said relationships are satisfied, and wherein said negotiator is operable to sum said variables to determine a satisfaction level for said objective.

162.The platform of claim 161, wherein said party goal program unit is operable to set a requirement of a minimum number of satisfied relationships for use by said negotiator.

163.A platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit for allowing each party to define a plurality of objectives in respect of said outcome, each objective comprising a plurality of variables, therefrom to form for each party a goal program, wherein a variable having discrete values is predefined, and wherein said party input unit is operable to receive discrete values of said variable for use in said objective definition,

a unifier, associated with said party goal program unit for receiving goal programs of respective parties, said objectives including discrete values of said variable, said unifier being operable to carry out unification of said goal programs by considering said discrete values to arrive at a common region of said discrete variables amongst said goal programs, and

a negotiator operable to utilize fulfillment levels associated with said discrete values to produce successive offers to converge on an outcome within said common region.

164.The platform of claim 163, wherein said party input unit is operable to accept weightings for respective discrete values of said variable, said weightings being usable to arrive at said fulfillment levels.

165.The platform of claim 164, wherein said party goal program unit is operable to use said discrete values and respective weightings to build summation functions, therefrom to express said variable in quantitative manner.

166.The platform of claim 165, wherein said party goal program unit is further operable to normalize said summation function by dividing by a largest one of said weightings.

167.The platform of claim 165, wherein said discrete values comprise Boolean variables, said Boolean variables serving as multipliers of respective weightings in said summation functions, wherein said negotiator is operable to reach said outcome by setting the Boolean variable of one of said discrete values to one and the remainder to zero, and then to calculate said summation functions, thereby to obtain a respective fulfillment level for each objective.

168.The platform of claim 167, wherein said unifier is operable to reattempt unification by setting Boolean variables of different ones of said discrete values to one, thereby to find a value of said variable which maximizes said fulfillment levels, for setting as said unified value.

169.The platform of claim 168, wherein said negotiator is operable to use a continuous variable as a transformation of said Boolean variables for carrying out said negotiating, said continuous variable being transformable back into said discrete Boolean variables to express said outcome.

170.The platform of claim 163, further comprising an offer delay timer, associated with said negotiator, for introducing determinable delays between issuance of successive offers to an opponent.

171.The platform of claim 170, wherein said offer delay timer is operable to set successively increasing delays.

172.The platform of claim 171, wherein said offer delay time is operable to set delays to change levelwise.

173.The platform of claim 170, wherein a magnitude of said delay is based on a relative change between succeeding opponent offers.

174.The platform of claim 170, wherein said offer delay timer is operable to set user determined delays.

175.A platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit comprising a party input unit for allowing a party to define at least one objective in respect of said outcome, and to associate said objective with at least two variables each having a target value, said party input unit further allowing input of a linkage between a first and a second of said variables, said linkage defining a trade-off path of deviations with respect to said target values, a unifier, associated with said goal program unit, said unifier being operable to use said series of variables to unify said at least one objective with other objectives to find a common area of interest, and

a negotiator, associated with said unifier, for using said trade-off path to find a mutually acceptable outcome within said common area.

176.The platform of claim 175, wherein said party goal program unit is operable to express said second variable as a function of said first variable, thereby to represent said linkage, and wherein said negotiator is operable to represent deviations from respective target values as deviations from the target value of said first variable.

177.The platform of claim 175, wherein said party goal program unit is operable to express said trade-off as separate deviation variables in respect of said first variable and in respect of said second variable wherein said separate deviation variables are orthogonal.

178.The platform of claim 175, wherein said party input unit is operable to permit an association of a relative importance level to said trade-off.

179.The platform of claim 175, wherein said party goal program unit is operable to calculate a relative importance level for said trade-off as an average of respective relative importance levels of said first and second variables.

180.The platform of claim 175, further comprising an offer delay timer, associated with said negotiator, for introducing determinable delays between issuance of successive offers to an opponent.

181.The platform of claim 180, wherein said offer delay timer is operable to set successively increasing delays.

182.The platform of claim 181, wherein said offer delay time is operable to set delays to change levelwise.

183.The platform of claim 180, wherein a magnitude of said delay is based on a relative change between succeeding opponent offers.

184.The platform of claim 180, wherein said offer delay timer is operable to set user determined delays.

185.A platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit for defining goal programs in respect of an outcome, the goal program unit comprising a party input unit for allowing a party to input data relating to said goal program, said goal program unit being operable to translate said values into objective functions and constraints on said objective functions within said goal program,

and a negotiator, associated with said goal program unit, said negotiator comprising an optimizer to find best values for said objective functions and constraints, therewith to obtain a best solution for the goal program for output as a first offer, and then iteratively to produce further solutions until an offer is accepted, thereby to achieve said outcome.

186.The platform of claim 185, wherein said negotiator comprises a percentage reducer for taking ones of said objective functions in turn, worsening them by a predetermined percentage, thereby to produce said further solutions.

187.The platform of claim 186, wherein said percentage reducer is settable to take each of said objective functions in turn beginning with a most important objective, until a solution is accepted.

188.The platform of claim 186, wherein said objective functions are arranged in levels and said percentage reducer is settable to take objective functions of a first of said levels only.

189.The platform of claim 185, wherein said negotiator comprises a worst case calculator for determining a worst case level for ones of said objective functions and constraints, thereby to obtain a worst acceptable offer.

190.The platform of claim 185, wherein said goal program is arranged as pairwise bounded variables, and said negotiator comprises an arbitrary case calculator for taking one of each pair of variables, setting it to an arbitrary value within its respective bounds, taking the other of said pair of variables and setting it to zero, therefrom to calculate ones of said iterative solutions.

191.The platform of claim 189,wherein said negotiator further comprises an average case calculator, associated with said optimizer and with said worst case calculator, for

taking said best solution and said worst solution, each solution having corresponding values,

associating ones of said corresponding values, and

constraining variables of said goal program towards an average of each of said corresponding values, therewith to provide an average solution.

192.The platform of claim 185, further comprising an offer delay timer, associated with said negotiator, for introducing determinable delays between issuance of successive offers to an opponent.

193.The platform of claim 192, wherein said offer delay timer is operable to set successively increasing delays.

194.The platform of claim 193, wherein said offer delay time is operable to set delays to change levelwise.

195.The platform of claim 192, wherein a magnitude of said delay is based on a relative change between succeeding opponent offers.

196.The platform of claim 192, wherein said offer delay timer is operable to set user determined delays.

197.The platform of claim 191, wherein said goal program objective functions are arranged levelwise and said average case calculator is operable to carry out said associating and said constraining successively levelwise, thereby to produce said series of iterative solutions.

198.The platform of claim 185, wherein said data input unit is operable to receive user defined output values, wherein said goal program unit is operable to set said output values as single value constraints and to flag said constraints as unchangeable for said unifier.

199.The platform of claim 198, wherein said data input unit is operable to output an error indicator if said single value constraints render said goal program insoluble.

200.A platform for supporting negotiation between parties to achieve an outcome, the platform comprising:

a party goal program unit for defining goal programs in respect of an outcome, the goal program unit comprising a party input unit for allowing a party to input values, said goal program unit being operable to translate said values into objective functions and constraints on said objective functions within said goal program,

and a negotiator, comprising a solution sorter for comparing goal program solutions by evaluation of said goal program for each one of a series of solutions and ranking the solutions according to said evaluations, said negotiator being operable to use said ranking to apply preference to different solutions.

201.The platform of claim 200, wherein said negotiator further comprises a thresholder associated with said solution sorter for applying a threshold to said evaluations to exclude ones of said series of solutions.

202.The platform of claim 201, wherein said thresholder further comprises a solution completer for applying best values to incompleted variables in incomplete ones of said solutions, thereby to allow said goal program to be evaluated for said incomplete solutions.

203. The platform of claim 200, wherein said solution sorter is settable to find the best solution from said series of solutions by identifying the highest ranked solution and discarding the remaining solutions.

204.The platform of claim 200, wherein said solution sorter comprises a memory set to hold a predetermined number of solutions, and a comparator to compare a new solution with each solution in said memory, and further comprising a control unit for adding said new solution to said memory if its evaluation is larger than any solution in said memory, and for discarding a lowest ranked solution in said memory.

205.The platform of claim 200, further comprising an offer delay timer, associated with said negotiator, for introducing determinable delays between issuance of successive offers to an opponent.

206.The platform of claim 205, wherein said offer delay timer is operable to set successively increasing delays.

207.The platform of claim 206, wherein said offer delay time is operable to set delays to change levelwise.

208.The platform of claim 205, wherein a magnitude of said delay is based on a relative change between succeeding opponent offers.

209.The platform of claim 205, wherein said offer delay timer is operable to set user determined delays.

210.A platform for supporting negotiation between a local party and an opponent party to achieve an outcome, the platform comprising a unifier, the unifier comprising:

a goal program input unit for receiving a local party's goal program and an opponent's goal program to be unified, said goal programs comprising objective functions associated with constraints and being arranged in levels,

an optimizer for finding best solutions to goal programs, connected to find best values for said objective functions and constraints of said local party's goal program levelwise, and

a worst case calculator for finding worst solutions for goal programs, connected to find worst values for said objective functions and constraints of said opponent's goal program levelwise,

said negotiator being operable to:

use said optimizer and said worst case calculator in succession level by level to produce successive best local and worst opponent value sets for evaluation therefrom to form level by level offers, and

to advance from one level to another level only following acceptance by said parties of a offer regarding a previous level.

211.The platform of claim 204, wherein said negotiator comprises a constraint updater for updating constraints upon advance from one level to another level in accordance with said respective acceptance.

212.The platform of claim 211, further comprising an offer improver operable to provide an improved offer by making a change in a selected one of said variables to bring about an improvement in the evaluation of the opponent's goal program.

213.The platform of claim 212, wherein said change in said variable is calculated such that said improvement is a predetermined proportion of a difference between a previous offer made and a best possible evaluation made for the opponent.

214.The platform of claim 213, wherein said offer improver is operable to use a value of said selected variable in a last opponent's offer to moderate said change.

215.The platform of claim 213, wherein said offer improver is operable to calculate a protection value, and to use said protection value to limit a reduction in the evaluation of the local party's goal program as a consequence of said improvement to the opponent's goal program evaluation.

216.The platform of claim 215, wherein said protection value is a proportion of the difference between a worst case evaluation of the local party's goal program and an evaluation of last previous offer thereof.

217.The platform of claim 210, further comprising an offer improver for taking goal program values of a previous local party offer and one value in turn from a previous opponent offer, testing the opponent value against local constraints, and if it fits within the constraints then substituting it into the previous local party offer thereby to provide an improved offer.

218.A platform for supporting negotiation between a local party and an opponent party to achieve an outcome, the platform comprising a negotiator, the negotiator comprising:

a goal program input unit for receiving a local party's goal program, said goal programs comprising objective functions associated with constraints and being arranged in levels,

an optimizer for finding best solutions to goal programs, connected to find best values for said objective functions and constraints of said local party's goal program levelwise, and

a stay close processor for determining variable improvement directions from monitoring of received offers from said opponent and carrying out value perturbations in said directions,

said negotiator being operable to:

use said optimizer to produce a first offer for a first level, to advance from one level to another level only following acceptance by said parties of a unification offer regarding a previous level, and

use said stay close processor to produce a first offer for each subsequent level.

219.The platform of claim 218, wherein said negotiator comprises a constraint updater for updating constraints upon advance from one level to another level in accordance with said respective acceptance.

220.The platform of claim 218, wherein said negotiator further comprises

a gap value determiner for determining a gap for use in offer improvement, and

a value improver, associated with said gap value determiner, for inserting a predetermined proportion of said gap as a constraint of said goal program, and wherein said stay close processor is associated with said value improver thereby to apply said predetermined gap proportion in said direction to provide an improved offer.

221.The platform of claim 220, wherein said stay close processor is operable to monitor two successive opponent offers for value changes therebetween, and to assign to each respective changing variable a weight for use in providing an improved offer, the magnitude of said weight being selected in accordance with a monitored relative size of a corresponding value change of said opponent.

222.The platform of claim 220, wherein said gap is a constant.

223.The platform of claim 222, wherein said constant is a difference between a best value and a worst value of a corresponding variable.

224.The platform of claim 220, wherein said gap is a difference between a local best case and a last opponent proposal.

225.The platform of claim 224, further comprising a gap truncator for maintaining said gap between predetermined maximum and minimum bounds.

226.The platform of claim 218, further comprising an offer delay timer, associated with said negotiator, for introducing determinable delays between issuance of successive offers to an opponent.

227.The platform of claim 226, wherein said offer delay timer is operable to set successively increasing delays.

228.The platform of claim 227, wherein said offer delay time is operable to set delays to change levelwise.

229.The platform of claim 226, wherein a magnitude of said delay is based on a relative change between succeeding opponent offers.

230.The platform of claim 226, wherein said offer delay timer is operable to set user determined delays.

231.A platform for joint processing of goal programs to produce an outcome, the platform comprising:

a party goal program unit for formulation of at least one local goal program,

a unifier for determining common ground between said local goal program and at least one other goal program,

a negotiation necessity tester, associated with said unifier, for joint solving of said local and said other goal program to form a joint goal program comprising optimal solutions for each of said local and said other goal program, said negotiation necessity tester being set to determine whether there lies a single solution that includes both optimal solutions within said common ground, and if so, to indicate that no negotiation is necessary.

232.A resource negotiator for making successive offers for usage of a resource with at least one remote party based on a goal program of a local party, the goal program comprising a plurality of objectives, at least one of said objectives having a target value, an upper bound, a lower bound and at least one constraint, the resource negotiator comprising:

an input for receiving data from said remote party,

a minimizer for producing successively worsening minimizations of said goal program, and

an offer formulator, associated with said minimizer, for formulating said minimizations into offers for resource usage for sending to said remote party.

233.The resource negotiator of claim 232, wherein said data from said remote party is a goal program comprising a plurality of objectives, at least some of said objectives having a target value, an upper bound, a lower bound and at least one constraint, said resource negotiator further comprising a maximizer for producing successively improving maximizations of said remote party goal program, for use together with said minimizations by said offer formulator for formulating said offers for resource usage.

234.The resource negotiator of claim 232, wherein said offer formulator comprises a behavior synthesizer for governing said offer formulation in accordance with a predetermined behavior profile.

235.The resource negotiator of claim 234, wherein said behavior profile comprises an opponent offer feedback feature for using opponent offer improvement levels for setting improvement levels for successive offer formulations.

236. The resource negotiator of claim 234, wherein said behavior profile comprises an opponent offer feedback feature for using opponent offer data for setting time intervals for successive offer outputs.

237.The resource negotiator of claim 234, wherein said behavior profile comprises a negotiation refusal condition for breaking off negotiations when said condition is achieved.

238.The resource negotiator of claim 237, wherein said refusal condition comprises a predetermined number of opponent offers that fail a predetermined improvement threshold.

239.The resource negotiator of claim 237, wherein said refusal condition comprises a predetermined time during which said negotiation fails to reach a predetermined improvement threshold.

240.The resource negotiator of claim 232, wherein said input is operable to receive data from a plurality of remote parties.

241.The resource negotiator of claim 232, wherein said offers are based on changes to one of said variables only.

242.The resource negotiator of claim 241, comprising a bound setter for setting at least one of respective upper and lower bounds of said variable as a function of bounds of other objectives in said goal program.

243.The resource negotiator of claim 241, comprising a dynamic bound setter for setting at least one of respective upper and lower bounds of said variable as a function of data received at said input.

244.A resource negotiator for negotiating for usage of a resource with a plurality of remote parties based on a goal program of a local party, the goal program comprising a plurality of objectives, at least one of said objectives having an upper bound, and a lower bound, the resource negotiator comprising:

an input for receiving data from said remote parties, an objective maximizer for maximizing a value of said at least one objective, and

an offer acceptor, associated with said maximizer, for receiving offers from said remote parties, comparing said maximizing with said offers and for accepting one of said offers based on said maximizations.

245.The resource negotiator of claim 244, wherein said offer acceptor is operable to accept an offer representing a least maximization.

246.The resource negotiator of claim 244, wherein said offer acceptor is operable to accept an offer representing a second least maximization.

247.The resource negotiator of claim 244, comprising an output for revealing received offers to each of said remote parties.

248.The resource negotiator of claim 244, comprising a bound setter for setting at least one of respective upper and lower bounds of said variable as a function of bounds of other objectives in a respective goal program.

249.The resource negotiator of claim 244, comprising a dynamic bound setter for setting at least one of respective upper and lower bounds of said variable as a function of data received at said input.

250.The resource negotiator of claim 244, wherein said maximizer is set to perform maximization over a plurality of objectives.

251.The resource negotiator of claim 250, wherein said maximizer is set to perform maximization over a penalty function of said objectives.

252.The resource negotiator of claim 251, further comprising a goal program output unit for sending to said plurality of remote parties at least some of said local party goal program objectives to enable said remote parties to evaluate potential offers in light thereof.

253.A resource negotiator for negotiating for usage of a resource with a plurality of remote parties based on a goal program of a local party, the goal program comprising at least one objective assignable with at least one of an upper bound, and a lower bound, the resource negotiator comprising:

an active bid monitor for monitoring remote parties remaining in said negotiating,

a value increaser for successively increasing a value of said at least one predetermined objective,

an offer acceptor, associated with said active bid monitor and with said value increaser, for ending said negotiation at a time at which only a predetermined number of remote parties remains active, and at a corresponding value of said at least one predetermined objective, said offer acceptor being operable to deem said negotiation successful if said corresponding value is within any assigned bounds, said predetermined number being related to a number of available resources.

254.The resource negotiator of claim 253, further comprising a bound assigner for calculating at least one bound of said predetermined variable according to other objectives of said local goal program.

255.The resource negotiator of claim 253, further comprising a goal program output unit for sending to said plurality of remote parties at least some of said local party goal program objectives to enable said remote parties to evaluate potential offers in light thereof.

256.The resource negotiator of claim 253, wherein said value increaser is operable to reduce said value to an intermediate level between a current and a previous level when a number of remaining parties drops from a level above said predetermined number to a level below said predetermined number, thereby to raise said number of remaining parties to correspond with said number of resources.

257.The resource negotiator of claim 256 wherein said predetermined number is one.

258.The resource negotiator of claim 253, wherein said active bid monitor comprises an output unit for revealing to said remote parties a number of remote parties remaining in said negotiating.

259.The resource negotiator of claim 253, wherein said active bid monitor comprises an output unit for revealing to said remote parties identities of remote parties remaining in said negotiating.

260.The resource negotiator of claim 253, wherein said predetermined number is one.

261.The resource negotiator of claim 253, further comprising a drop out decision unit for use by one of said remote parties, said unit comprising:

a current offer evaluator for expressing a current value issued by said value increaser in terms of a goal program of said remote party,

an active bid unit for storing the number of remote parties remaining in said negotiating,

a drop out function for calculating a drop out probability as a function of said current value and said number of remote parties remaining in said negotiating, and

a decision maker for using said drop out probability to decide whether to leave said negotiating.

262.A platform for performing ranking between database entries, each of said entries comprising a series of values arranged in fields, the platform comprising:

a trade-off unit for taking values from a plurality of fields of respective entries, defining a trade-off relationship therebetween such that for any given combination of values in said fields a single trade-off value is defined, and

a ranking unit for performing ranking amongst said entries in accordance with a respective single trade-off value.

263.The platform of claim 262, wherein said trade-off unit is operable to take said values in twos, such that each trade-off value is a trade-off between two fields.

264.The platform of claim 263, wherein said trade-off value comprises a reduction in a first of said variables being traded for a proportional increase in a second of said variables.

265.The platform of claim 262, wherein said trade-off unit is operable to take said values in groups of three or more.

266.The platform of claim 262,wherein said trade-off unit is operable to take said values in a plurality of trade-off groups, and to compile a separate trade-off statement for each group.

267.The platform of claim 266, wherein each trade-off statement comprises compatible variables.

268.The platform of claim 266, wherein said ranking unit is operable to determine an average of the deviations for each trade-off statement for use in said ranking.

269.The platform of claim 262, wherein said trade-off relationship comprises an inequality between corresponding values of successive entries.

270.The platform of claim 262, wherein said trade-off unit further comprises a weight assigner for assigning weights to fields, and wherein said trade-off relationship comprises assignment of a weight to each of said fields using said weight assigner, therewith to perform summation over each of said entries.

271.The platform of claim 270, wherein said weight assigner comprises a user preference input for receiving a user defined preference between ones of said entries, and wherein said weight assigner is operable to select weights for assignment to fields of said entries such as to enforce said user defined preference.

272. The platform of claim 271, wherein said weight selection is such as to maximize said user preferences.

273. The platform of claim 270, wherein said fields are ordered preferentially and wherein said weight assigner is operable to assign weights according to a position of a respective field in said order.

274.The platform of claim 271, wherein said weight assigner comprises a user input for receiving a parameter defining a number of entries of high desirability from the start of an ordered list, said weight assigner being operable to select weights to reinforce said desirability.

275.The platform of claim 262, comprising a ranking expression unit for providing an expression basis for defining at least one of a group comprising a condition, a deviation, a deviation condition, a constraint, a simple trade-off relationship, a complex trade-off relationship, a preferred value within a range, a preferred range, a weighting, therefrom to form an objective function for use in said ranking.

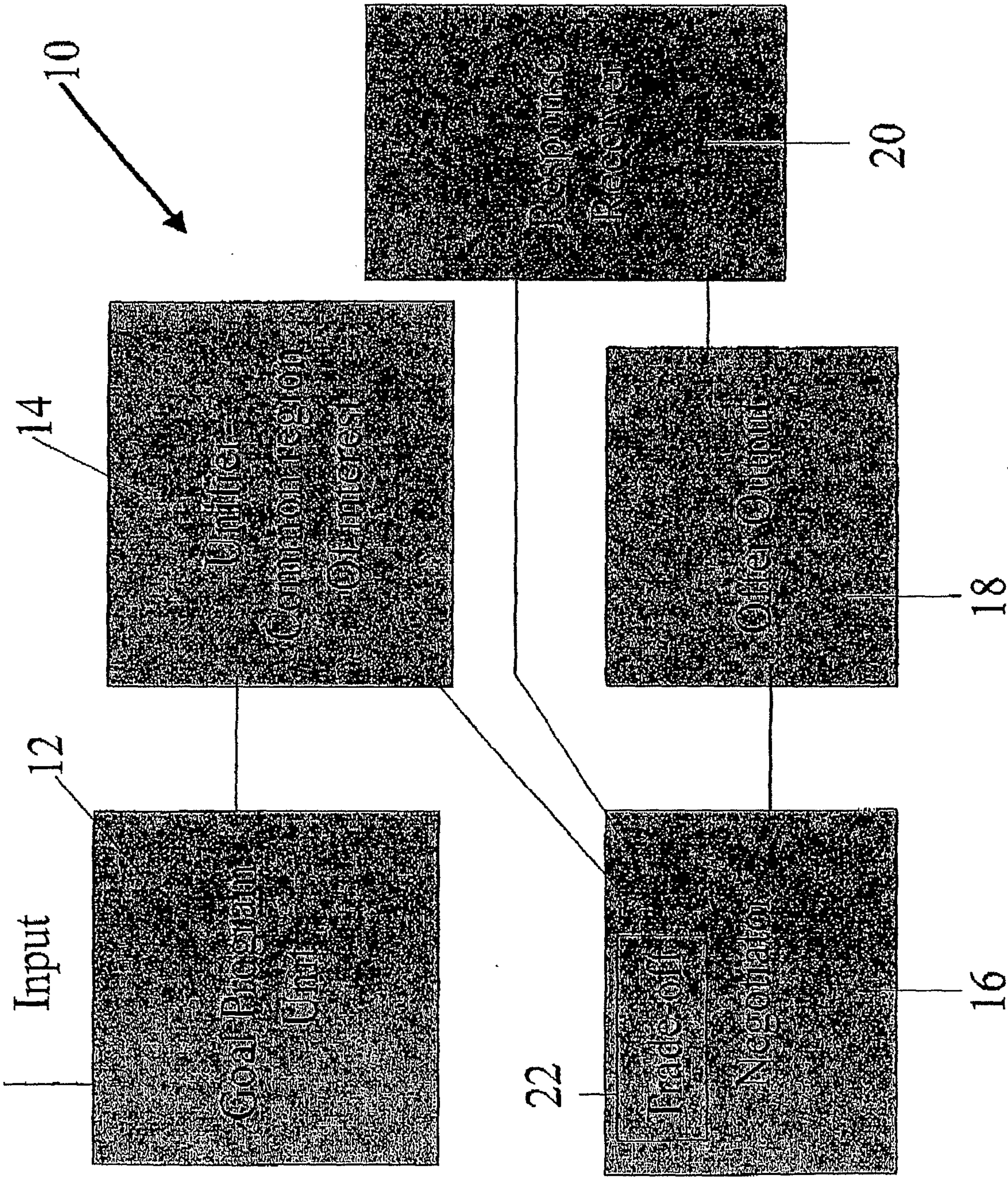

10
Input
Goal Processing Unit
12
Unifier - Common region Of interest
14
Response Receiver
20
Offer Output
18
Trade-off
Negotiation
22
16

Fig. 1

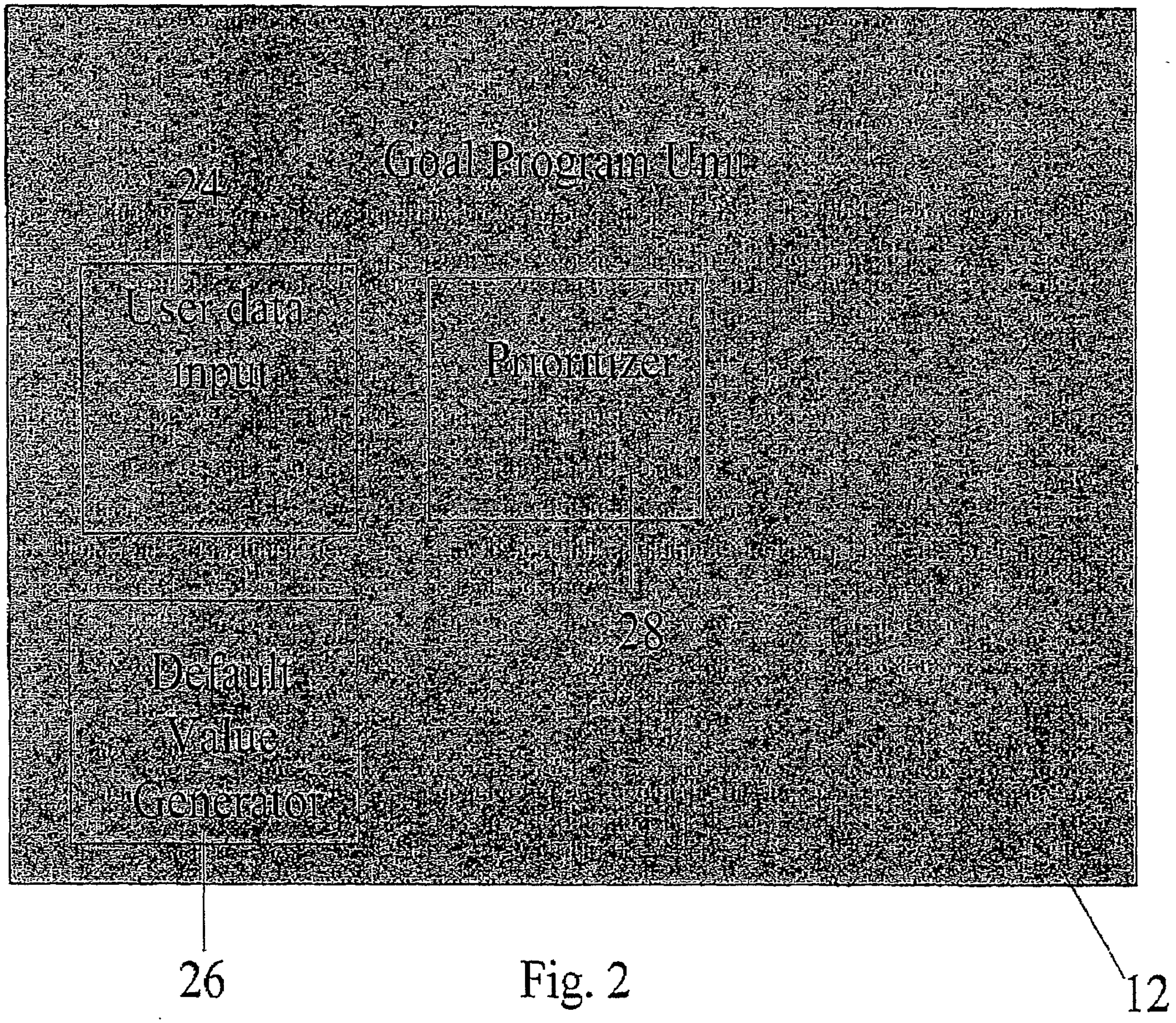

Goal Program Unit
24
User data input
Prioritizer
28
Default Value Generator
26
12

Fig. 2

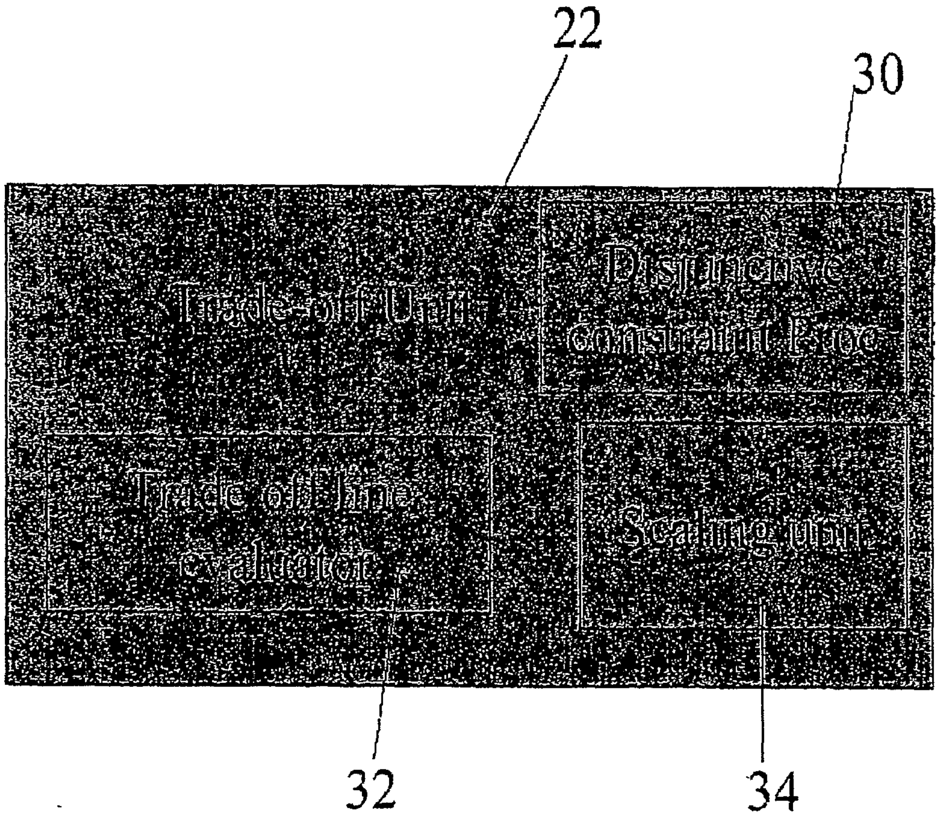

22
30
Trade-off Unit
Disjunctive constraint Proc
Trade off line evaluator
Scaling unit
32
34

Fig. 3

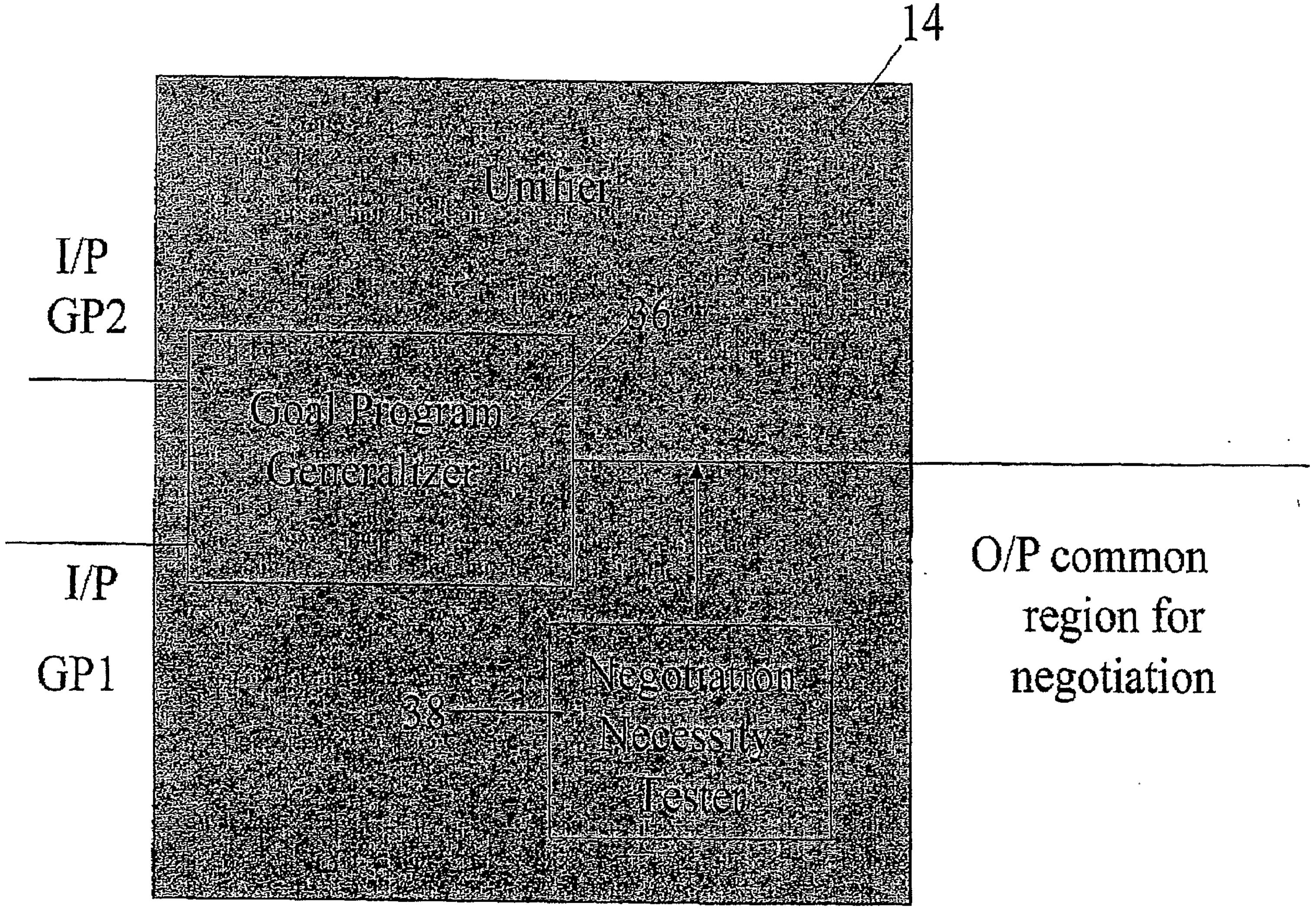

14
Unifier
I/P
GP2
36
Goal Program
Generalizer
I/P
GP1
38
Negotiation
Necessity
Tester
O/P common
region for
negotiation

Fig. 4

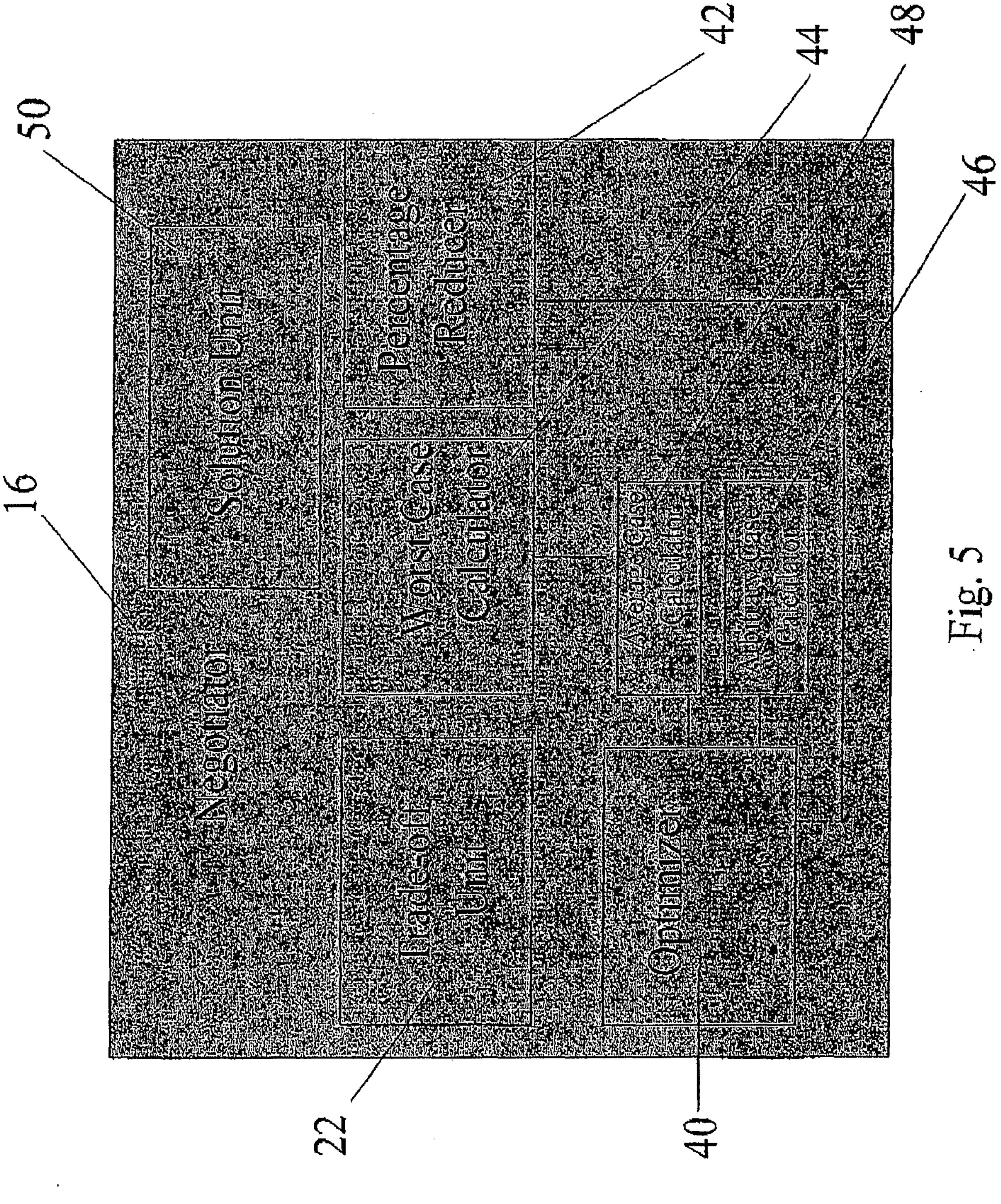

50
16
Solution Unit
Negotiator
42
44
48
46
Percentage Reducer
Worst Case Calculator
Trade-off Unit
Average Case Calculator
Arbitrary Case Calculator
Optimizer
22
40

Fig. 5

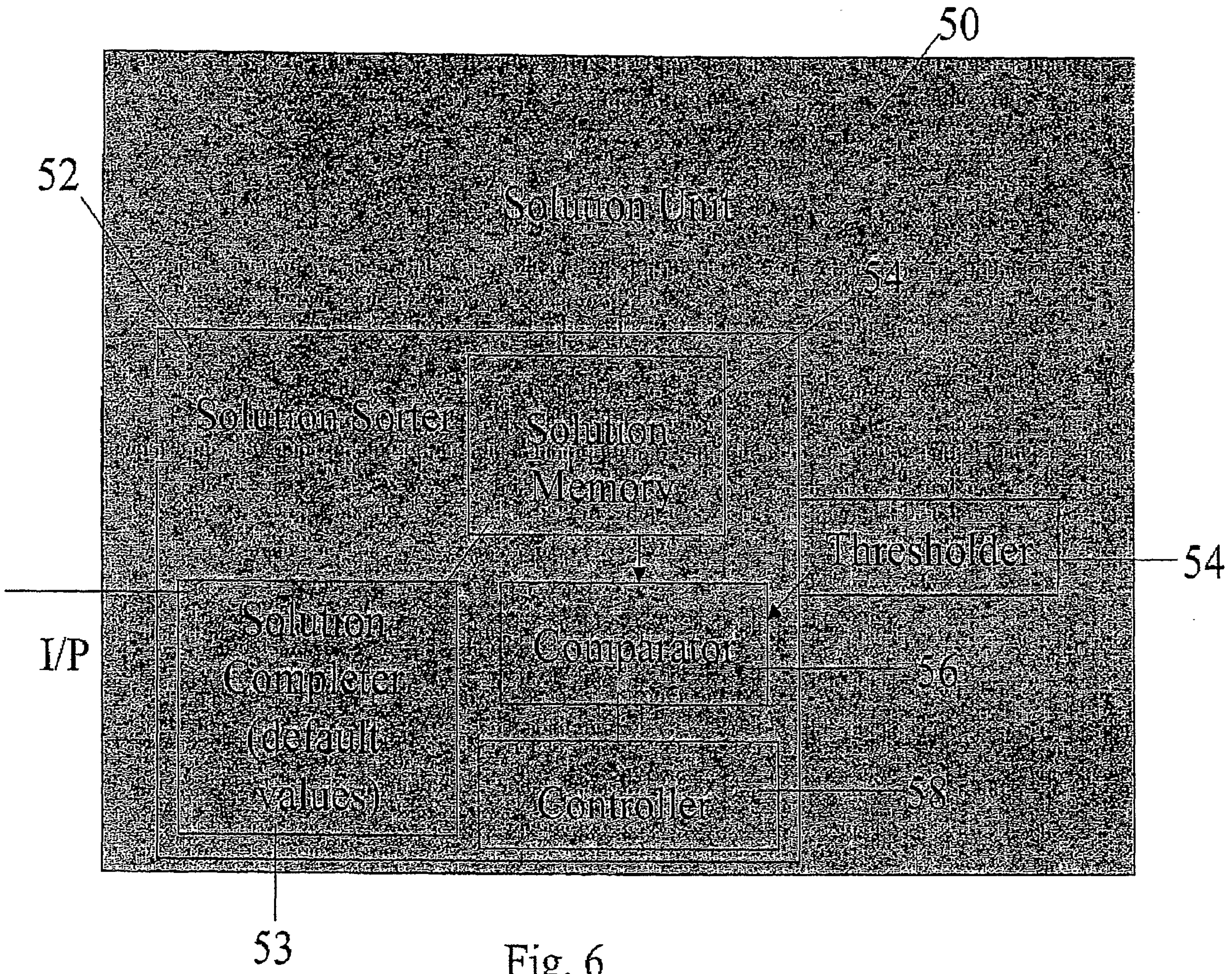

50
52
Solution Unit
54
Solution Sorter
Solution Memory
Thresholder
54
I/P
Solution Completer (default values)
Comparator
56
Controller
58
53

Fig. 6

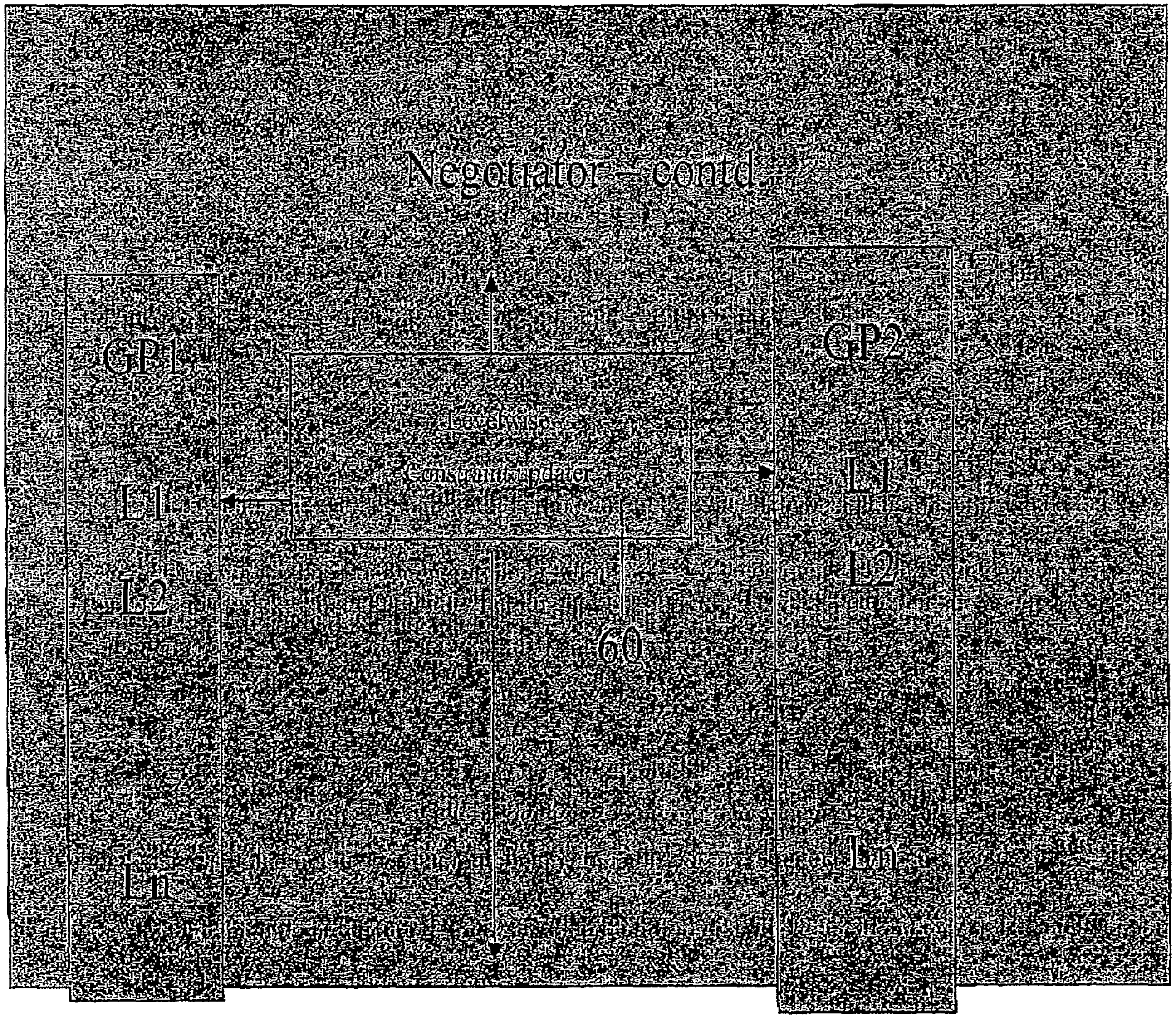

Negotiator – contd.
GP1
L1
L2
Ln
Levelwise
Constraint updater
60
GP2
L1
L2
Ln

Fig. 7

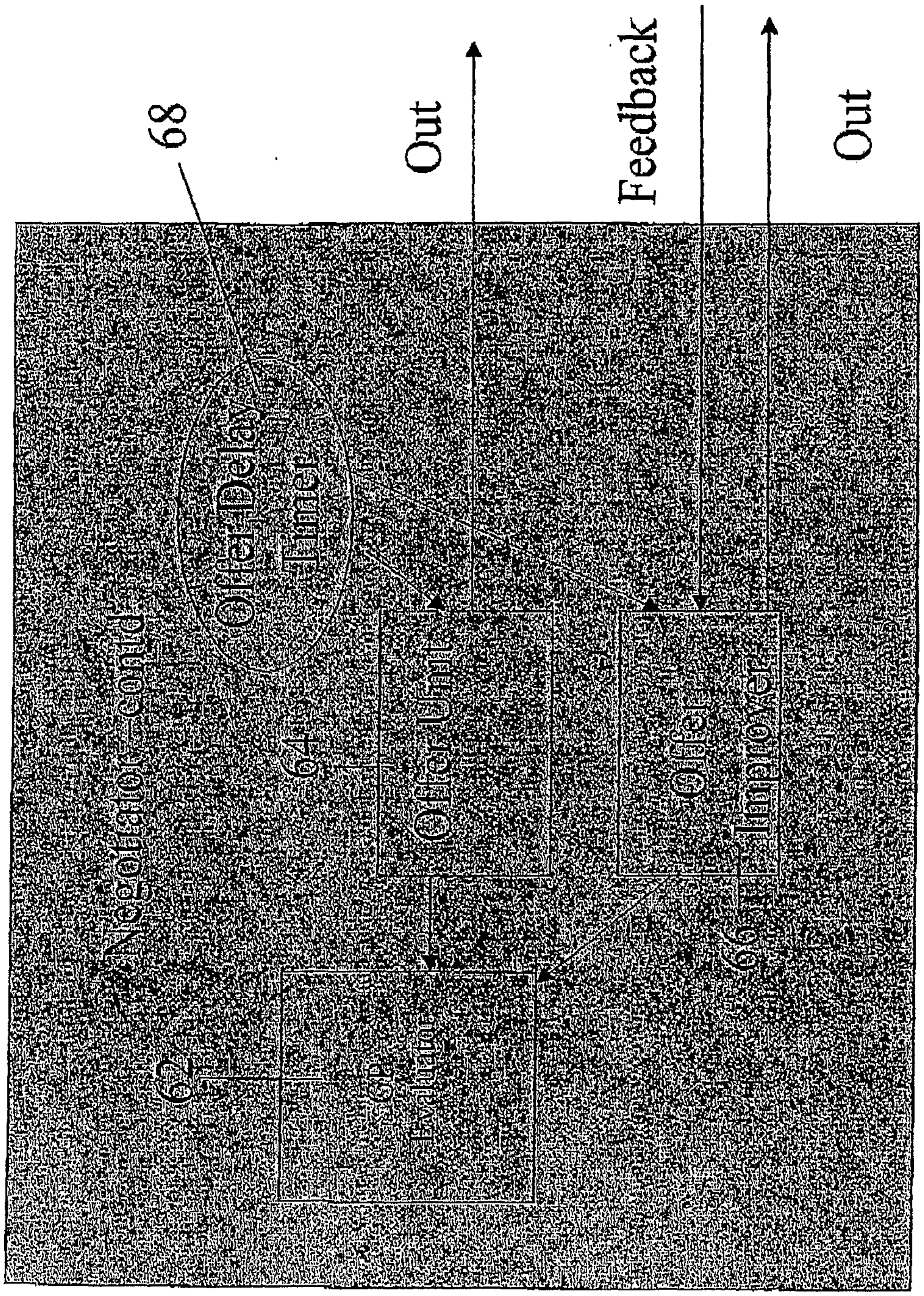

68
Out
Feedback
Out
Negotiator-contd
Offer Delay Timer
64
Offer Unit
Offer Improver
66
62
Evaluator

Fig.8

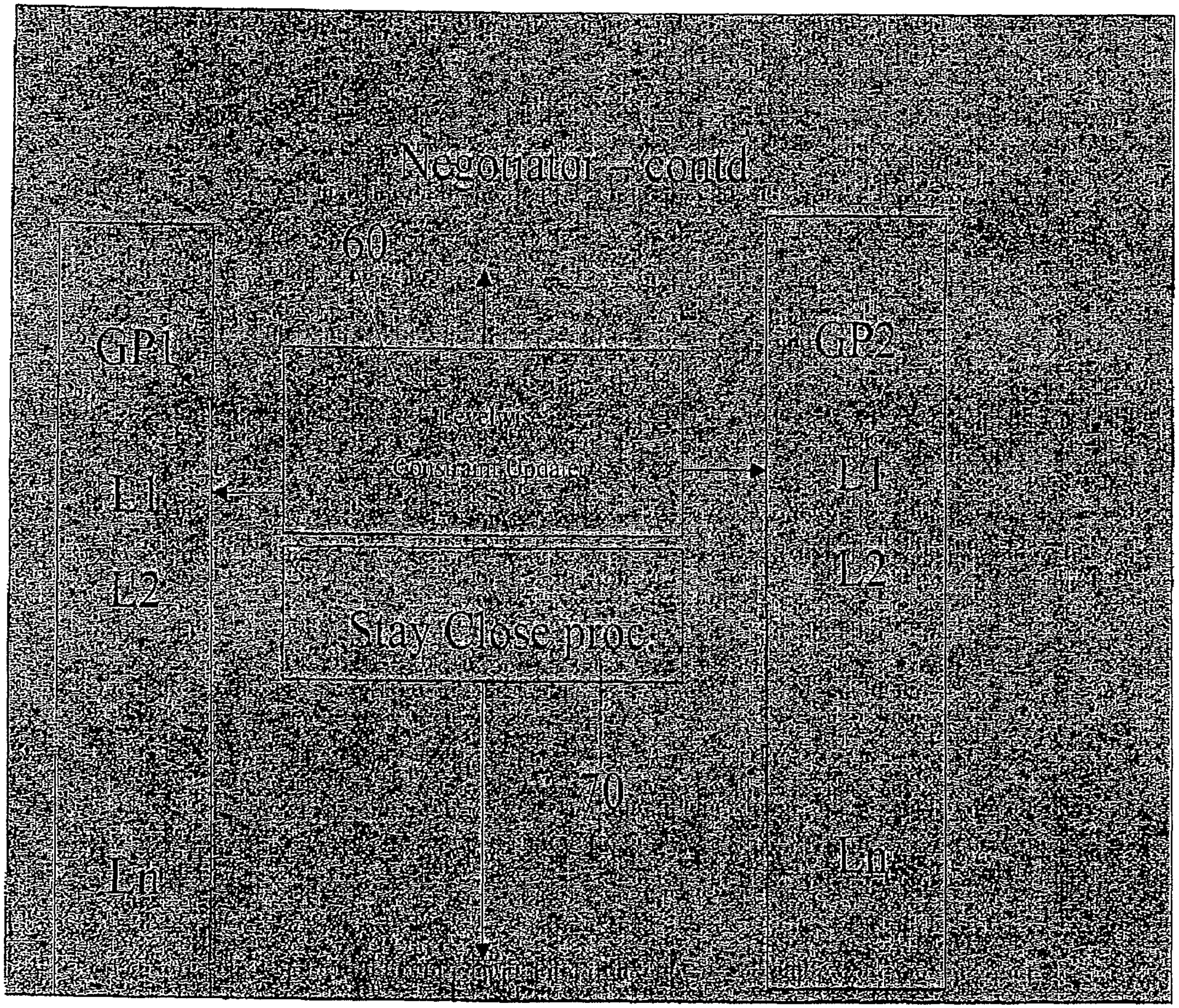

Negotiator – contd
60
GP1
L1
L2
Ln
Level
Constraint Update
Stay Close proc.
70
GP2
L1
L2
Ln

Fig. 9

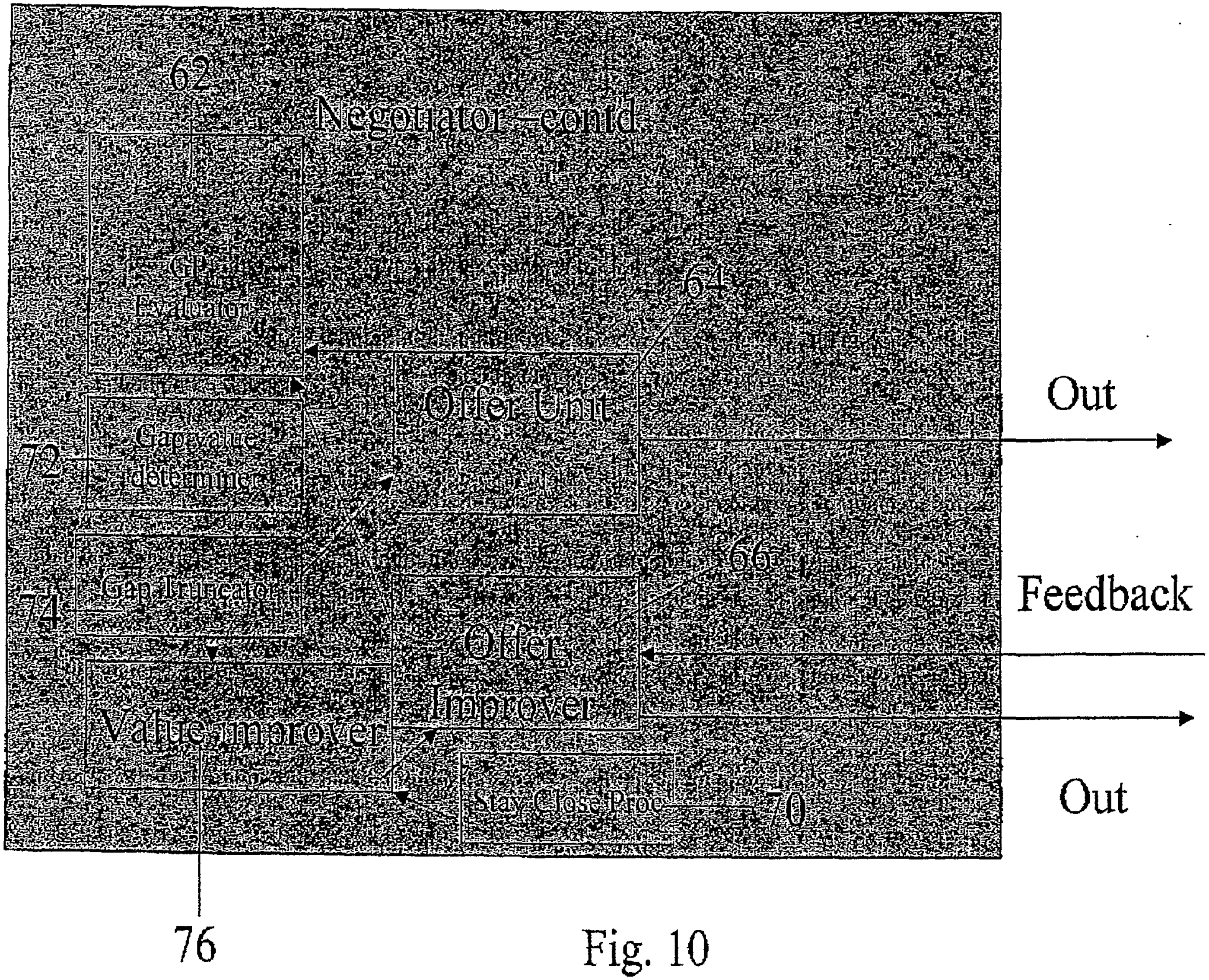

62
Negotiator –contd
GP
Evaluator
64
Offer Unit
Out
72
Gap value
determiner
74
Gap Truncator
66
Offer
Improver
Feedback
Value improver
Stay Close Proc
70
Out
76

Fig. 10

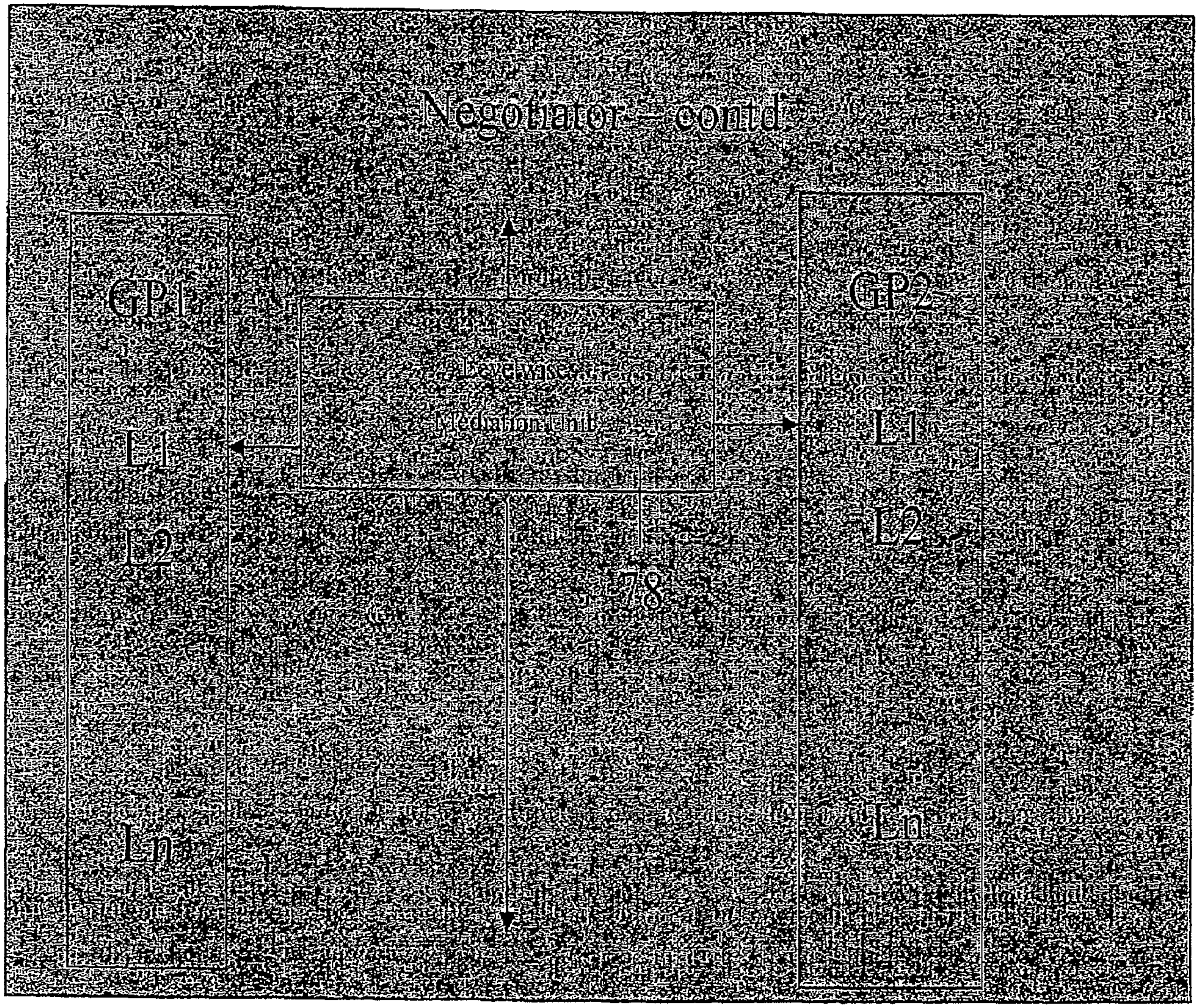

Negotiator – contd
GP1
L1
L2
Ln
Levelwise
Mediation Unit
78
GP2
L1
L2
Ln

Fig. 11

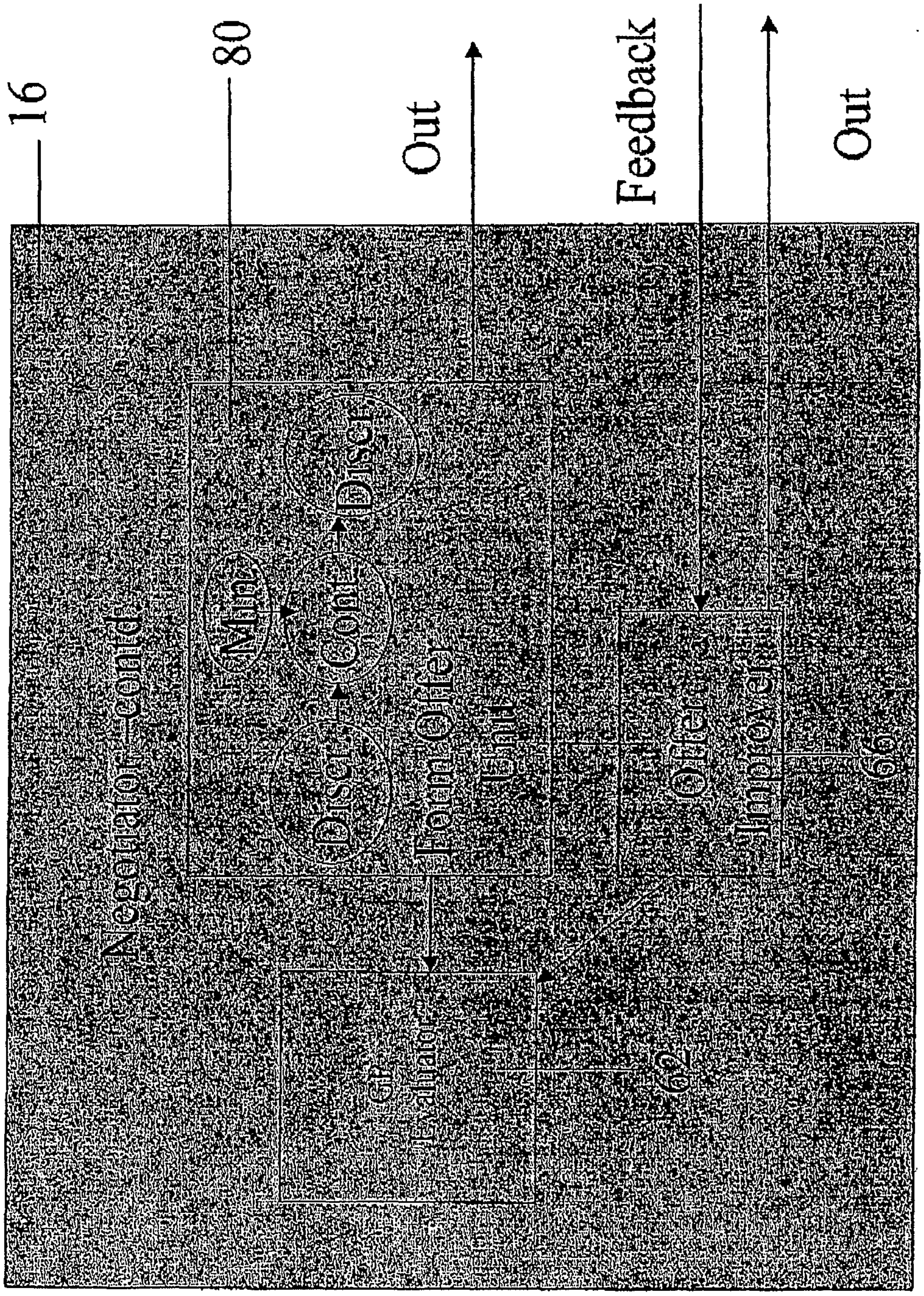

16
80
Out
Feedback
Out
Negotiator–cont'd
Min
Disc
Cont
Disc
Form Offer Unit
GP
Evaluator
62
Offer Improver
66

Fig. 12

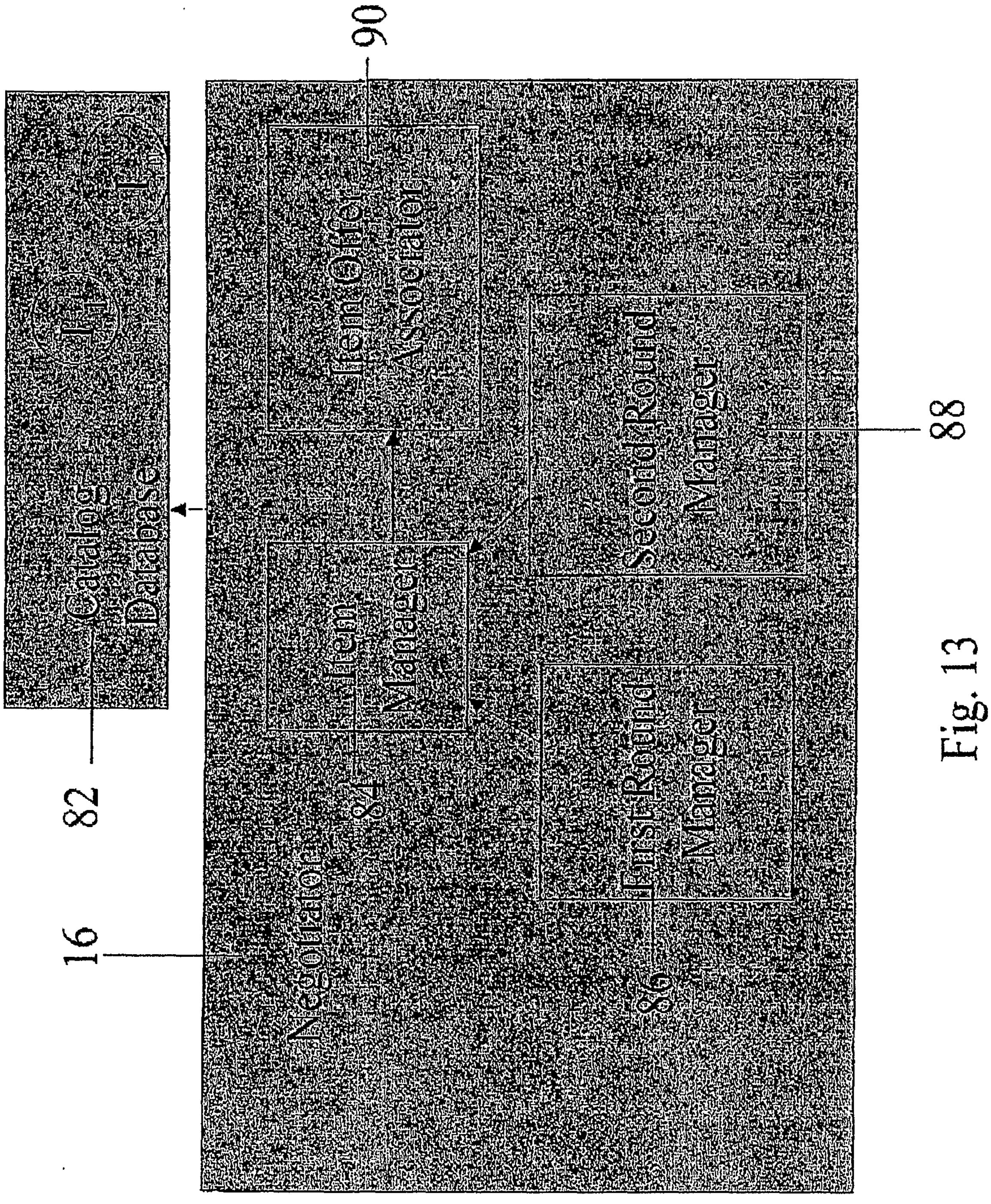

16
Negotiator
82
Catalog Database
I1
I
84
Item Manager
90
Item Offer Associator
86
First Round Manager
88
Second Round Manager

Fig. 13

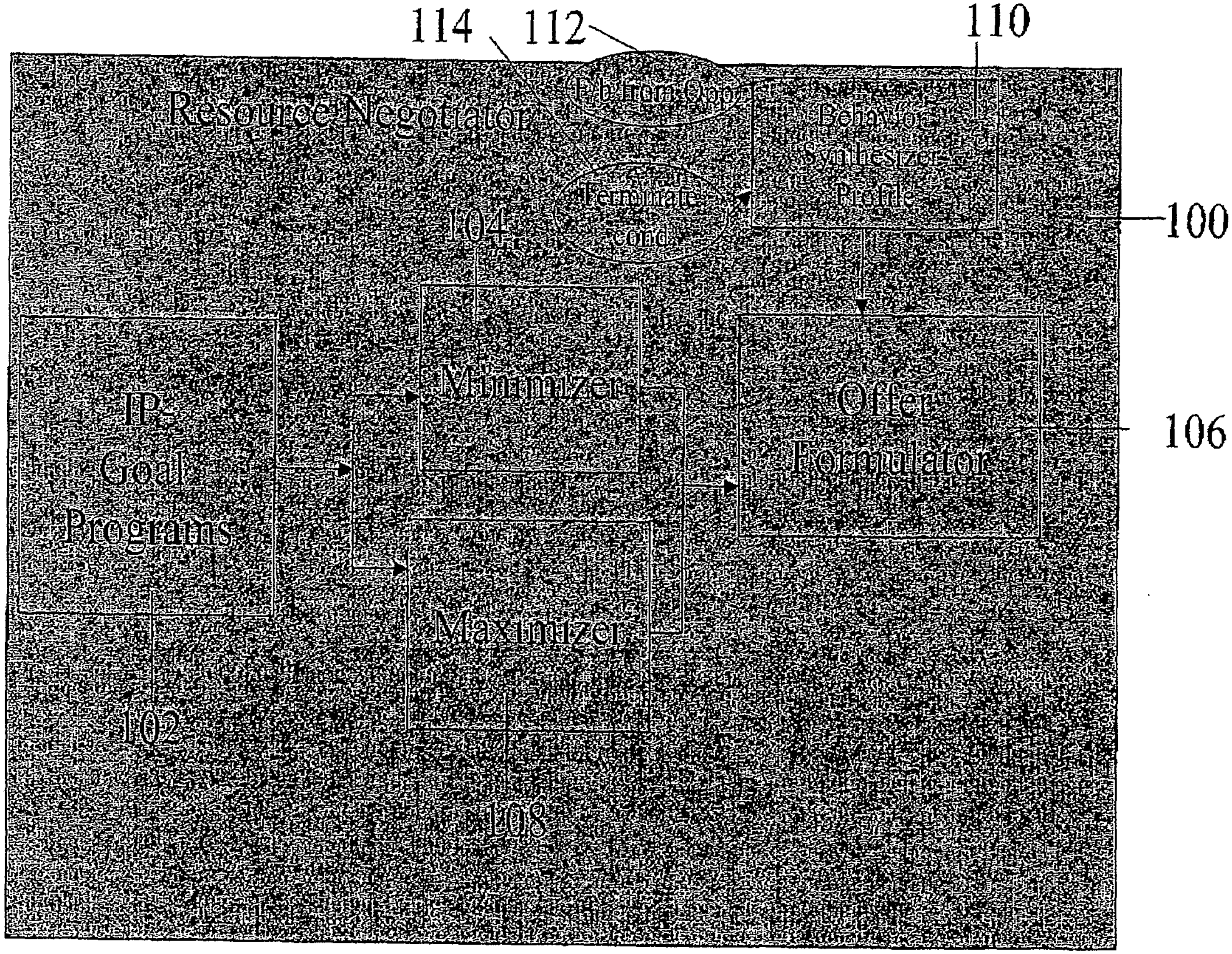
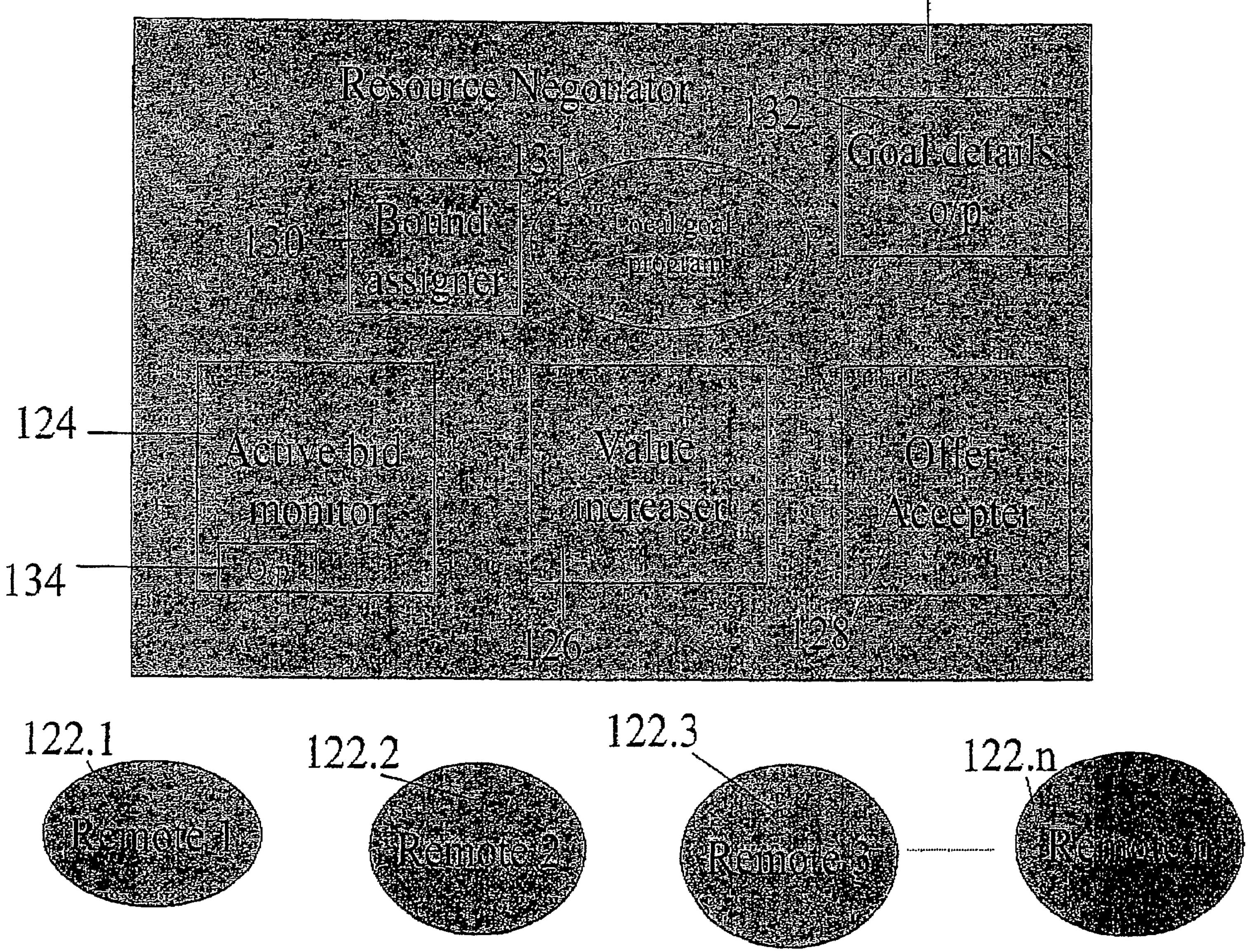

114
112
110
Resource Negotiator
Fb from Opp
Behavior Synthesizer Profile
Terminate cond
100
104
Minimizer
IP Goal Programs
Offer Formulator
106
Maximizer
102
108

Fig. 14

Fig. 15


120
Resource Negotiator
132
131
130
Bound assigner
Local goal program
Goal details o/p
124
Active bid monitor
134
o/p
Value increaser
Offer Accepter
126
128
122.1
Remote 1
122.2
Remote 2
122.3
Remote 3
122.n
Remote n

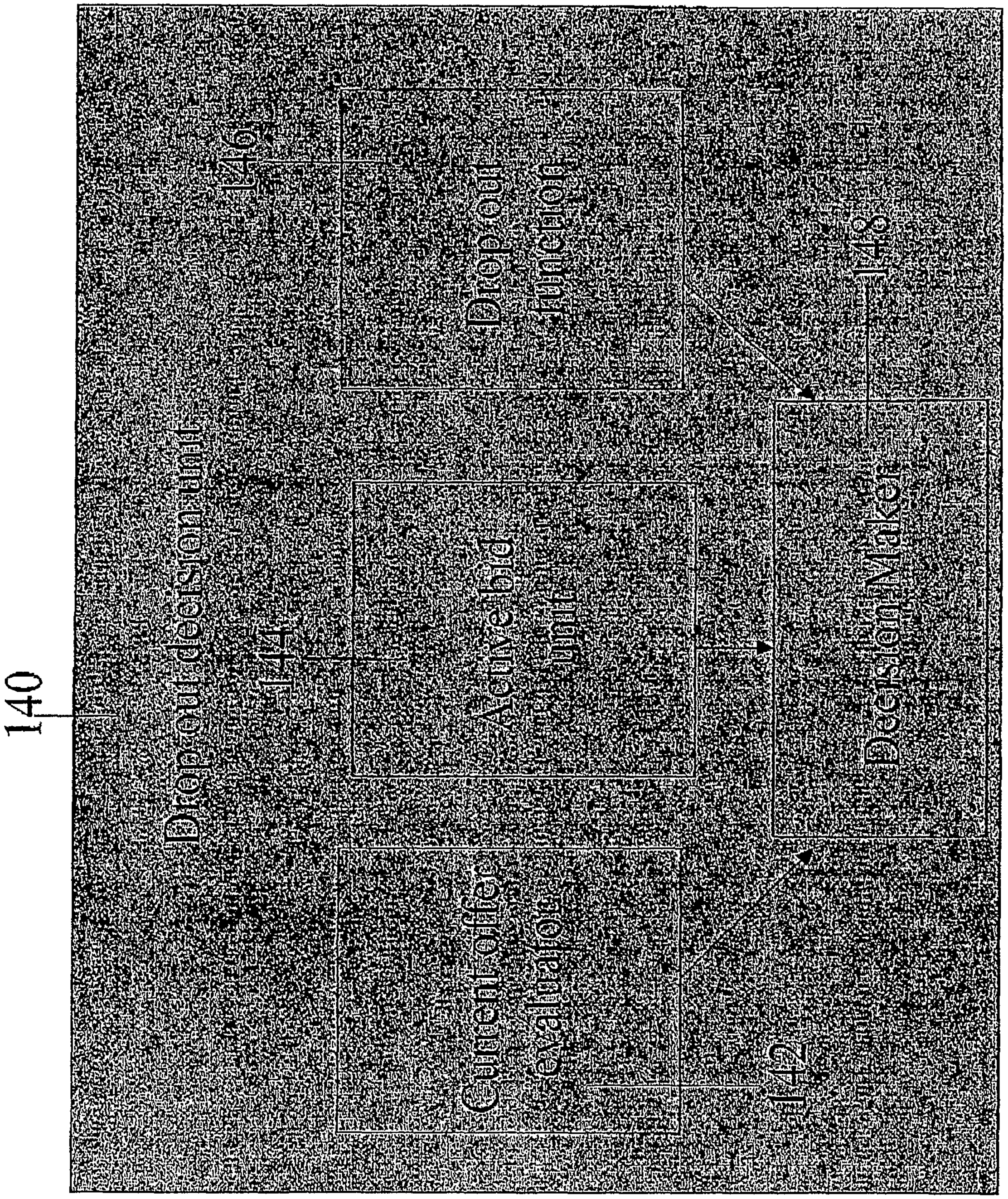

140
Dropout decision unit
146
Dropout function
144
Active bid unit
148
Decision Maker
142
Current offer evaluation

Fig. 16

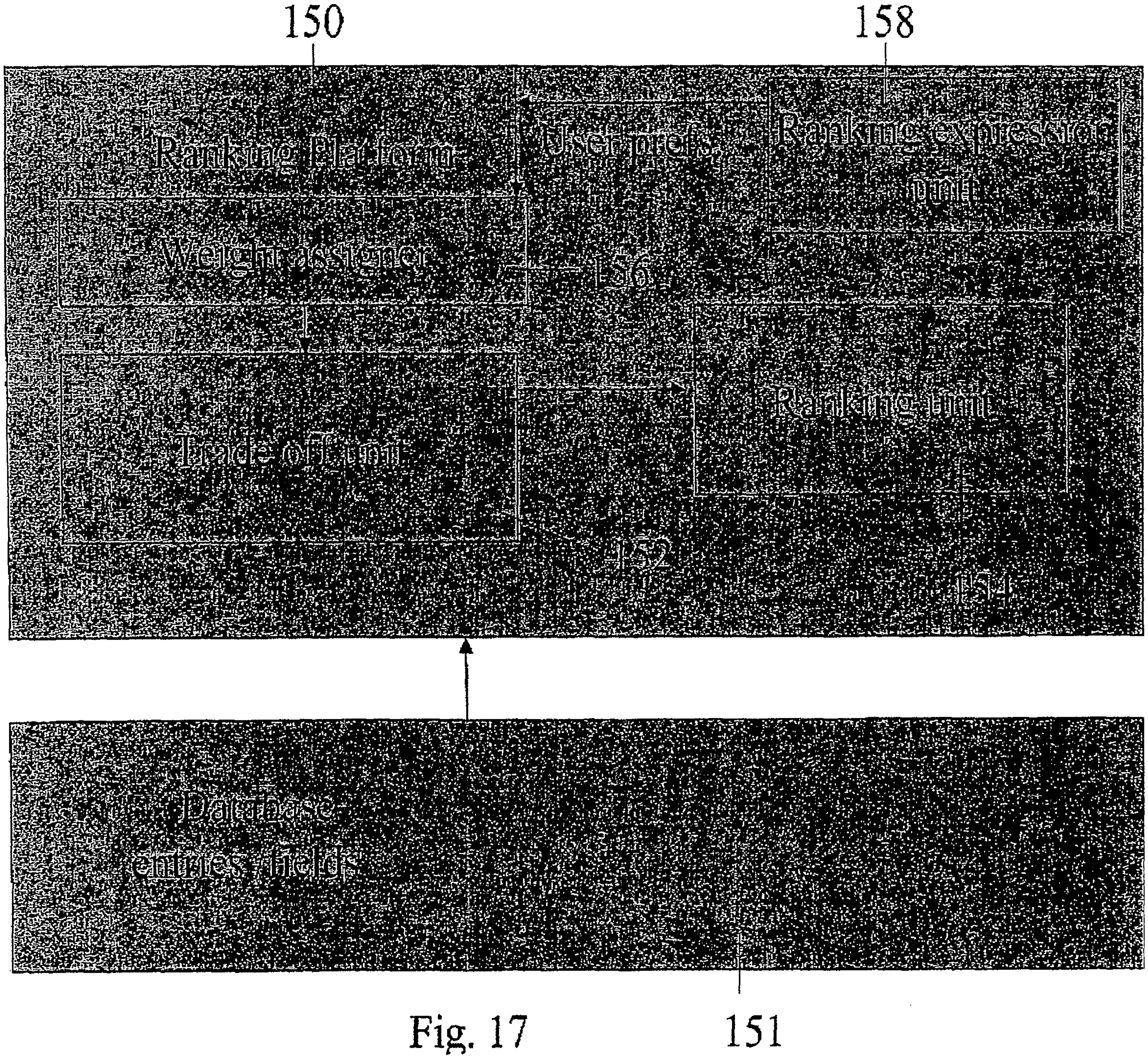

150
158
Ranking Platform
User prefs.
Ranking expression unit
Weight assigner
156
Trade off unit
Ranking unit
152
154
Database entries, fields
151

Fig. 17

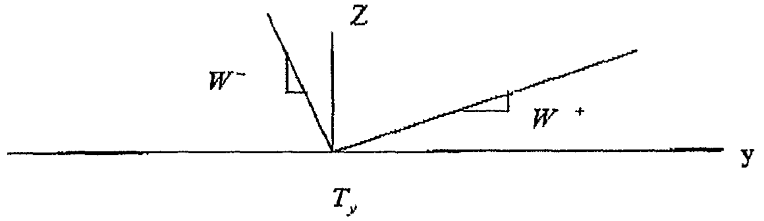


Fig. 18

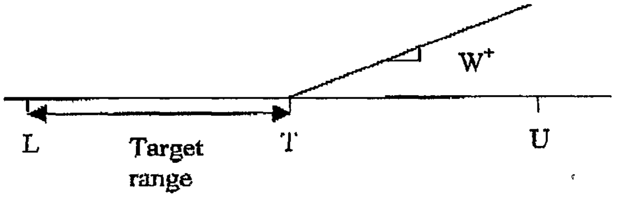

W^+
L
Target range
T
U

Fig. 19

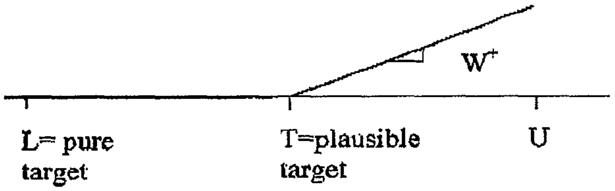

W^+
L= pure target
T=plausible target
U

Fig. 20

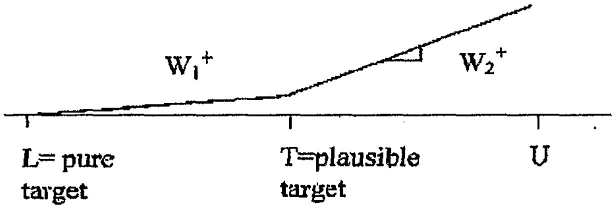

W_1^+
W_2^+
L= pure target
T=plausible target
U

Fig. 21

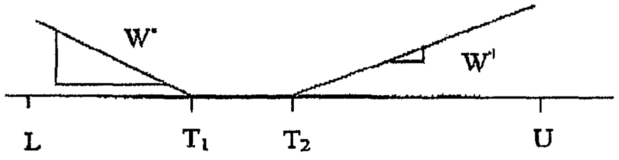
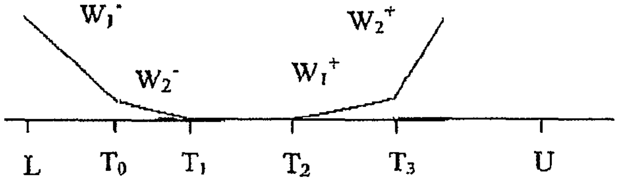
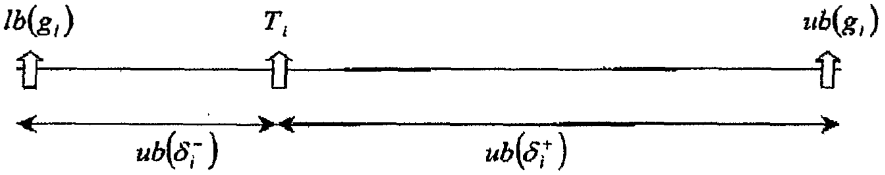


Fig. 22



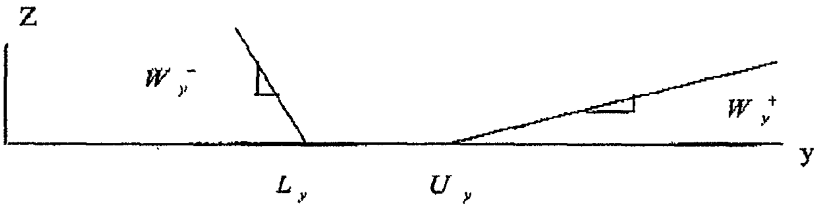
W_1^-
W_2^-
W_1^+
W_2^+
L
T_0
T_1
T_2
T_3
U

Fig. 23



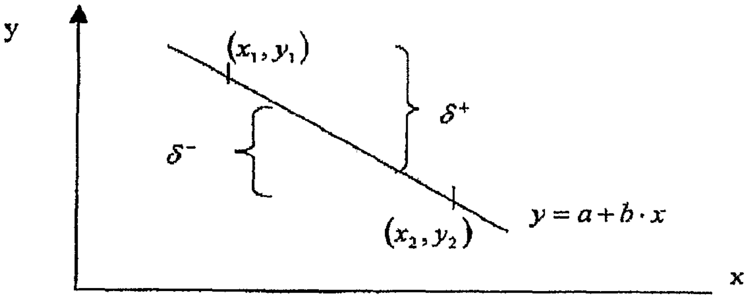
$lb(g_i)$
T_i
$ub(g_i)$
$ub(\delta_i^-)$
$ub(\delta_i^+)$

Fig. 24



Z
W_y^-
W_y^+
y
L_y
U_y

Fig. 25



y
(x_1, y_1)
δ^+
δ^-
(x_2, y_2)
$y = a + b \cdot x$
x

Fig. 26

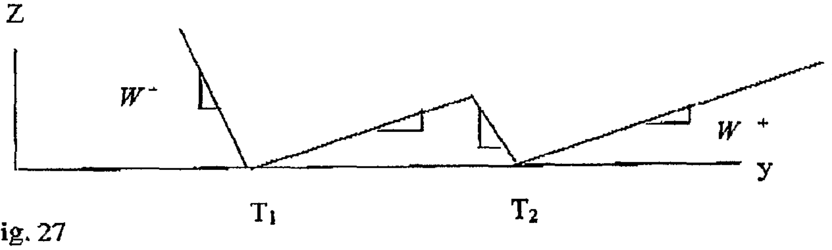

Z
W^-
W^+
y
T_1
T_2

Fig. 27

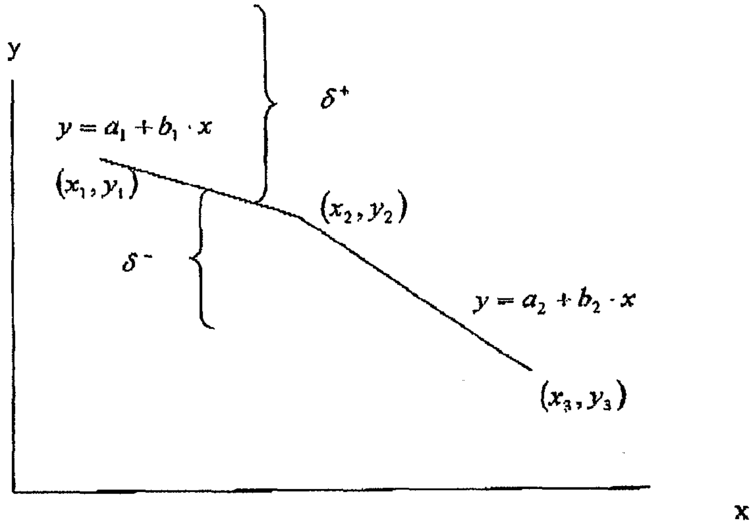

y
δ^+
$y = a_1 + b_1 \cdot x$
(x_1, y_1)
(x_2, y_2)
δ^-
$y = a_2 + b_2 \cdot x$
(x_3, y_3)
x

Fig. 28

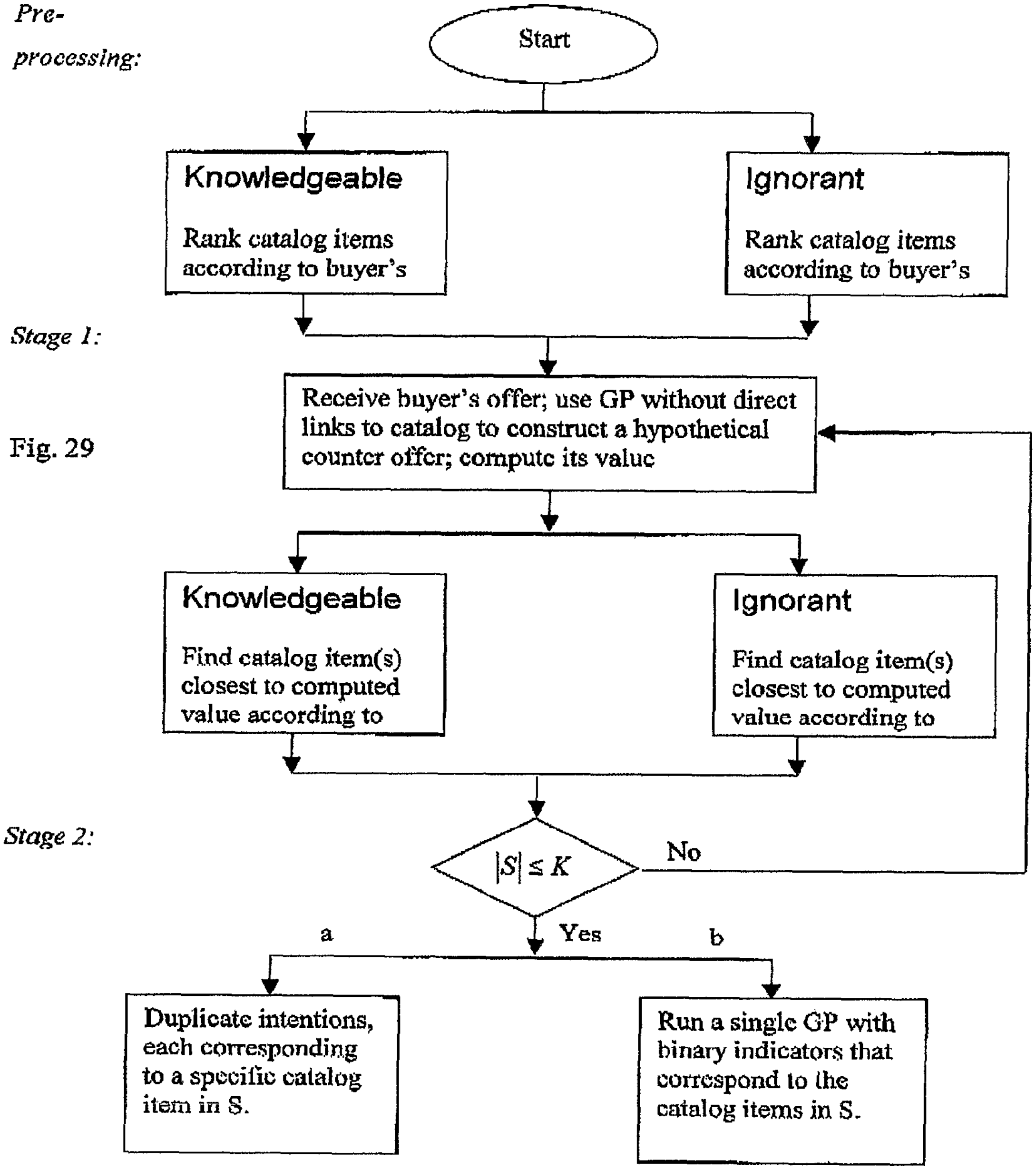

Pre-processing:
Start
Knowledgeable
Rank catalog items according to buyer's
Ignorant
Rank catalog items according to buyer's
Stage 1:
Receive buyer's offer; use GP without direct links to catalog to construct a hypothetical counter offer; compute its value
Knowledgeable
Find catalog item(s) closest to computed value according to
Ignorant
Find catalog item(s) closest to computed value according to
Stage 2:
$|S| \leq K$
No
Yes
a
b
Duplicate intentions, each corresponding to a specific catalog item in S.
Run a single GP with binary indicators that correspond to the catalog items in S.

Fig. 29

Fig. 30

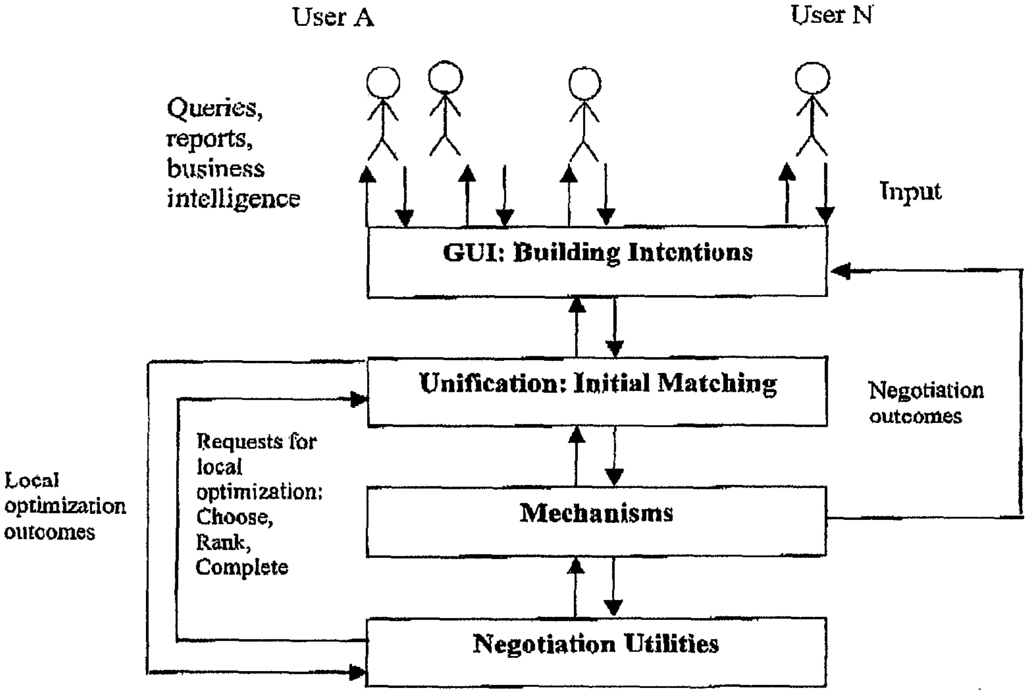

User A
User N
Queries, reports, business intelligence
Input
GUI: Building Intentions
Unification: Initial Matching
Negotiation outcomes
Requests for local optimization: Choose, Rank, Complete
Local optimization outcomes
Mechanisms
Negotiation Utilities

Fig. 31

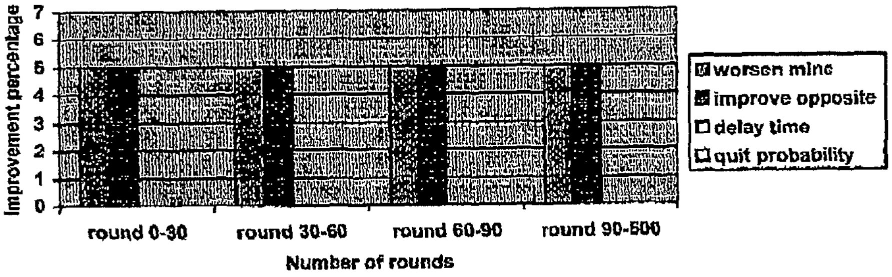


Fig. 32

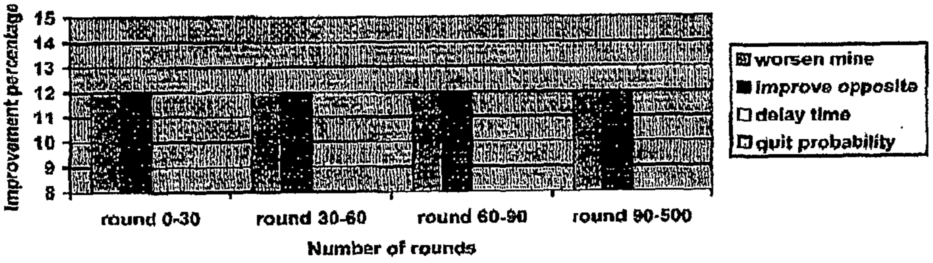
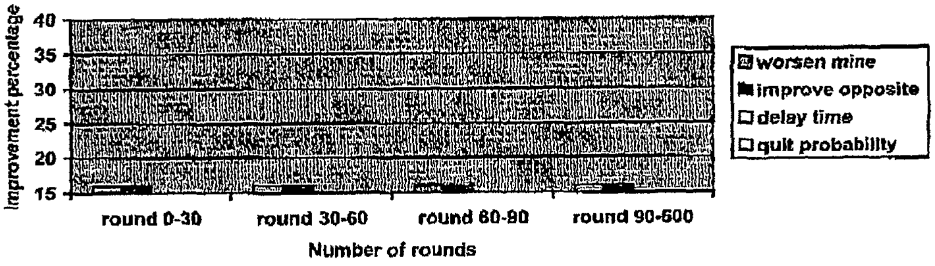


Fig. 33



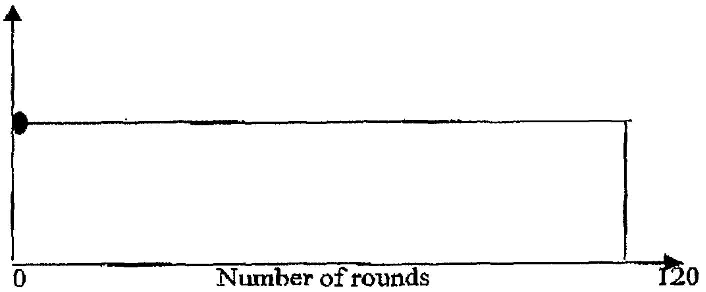


Fig. 34

Fig. 35

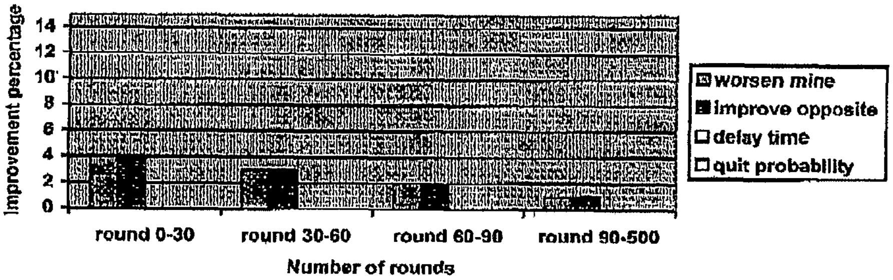


Fig. 36

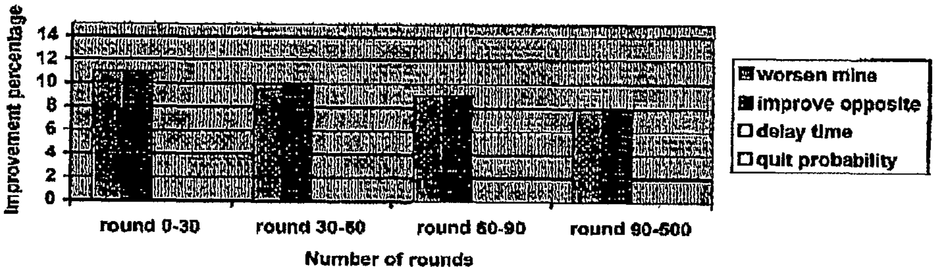


Fig. 37

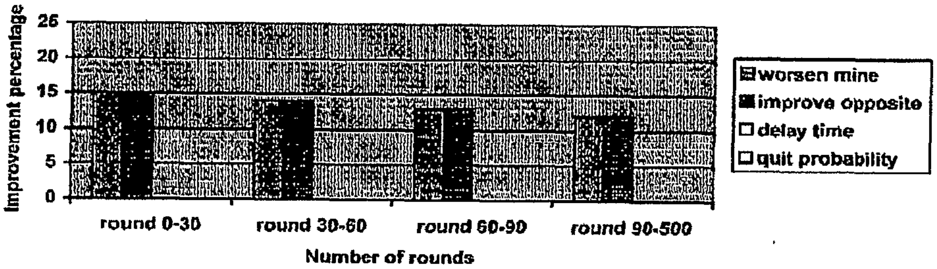


Fig. 38

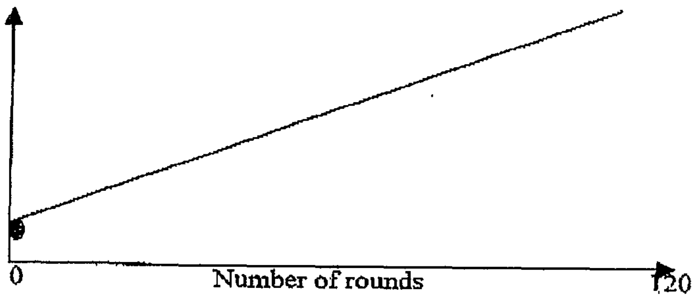


Fig. 39

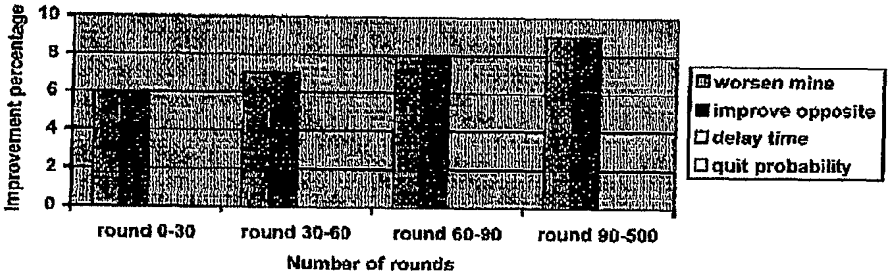


Fig. 40

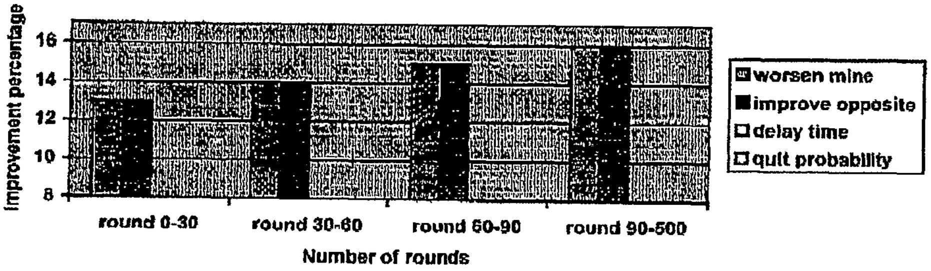


Fig. 41

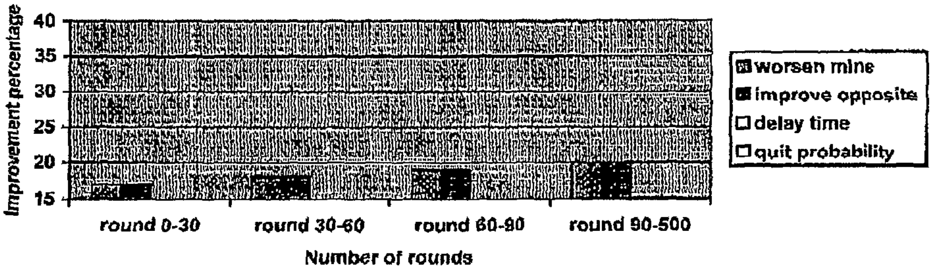

Improvement percentage
40
35
30
25
20
15
▨ worsen mine
■ improve opposite
☐ delay time
☐ quit probability
round 0-30
round 30-60
round 60-90
round 90-500
Number of rounds

Fig. 42

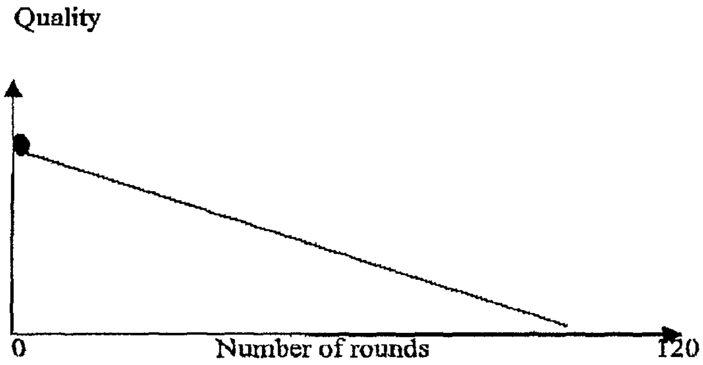

Quality
0
Number of rounds
120

Fig. 43